TEK SPECTRUM ANALYZERS

Part No. 070-8132-01
Product Group 2E

# THE 2711 & 2712 SPECTRUM ANALYZERS

# Programmer Manual

Copyright © 1991 by Tektronix, Inc., Beaverton, Oregon. Printed in the United States of America. All rights reserved. Contents of this publication may not be reproduced in any form without permission of Tektronix, Inc.

U.S. and foreign Tektronix products covered by U.S. and foreign patents and/or patents pending.

TEKTRONIX is a registered trademark of Tektronix, Inc.

Specification and price change privileges reserved.

***Please check for CHANGE INFORMATION at the rear of this manual.***

First Printing DEC 1991
Revised Feb 1992

Tektronix
COMMITTED TO EXCELLENCE

# TABLE OF CONTENTS

**Table of Contents** ..... i
**Operators Safety Summary** ..... v
TERMS IN THIS MANUAL ..... v
TERMS MARKED ON EQUIPMENT ..... v
AC POWER SOURCE ..... vi
DC POWER SOURCE ..... vi
PRODUCT GROUNDING ..... vi
INPUT POWER AND VOLTAGE LIMITATIONS ..... vii
USE THE PROPER FUSE ..... vii
GENERAL PRECAUTIONS ..... vii

**Section 1 — Introduction**
GPIB OPERATION ..... 1-1
OPERATION OVER THE GPIB ..... 1-4
System Controller ..... 1-4
Software Device Driver ..... 1-4
2711 and 2712 Equipped with the GPIB ..... 1-5
Interconnecting Cable ..... 1-5
Application Software ..... 1-6
Printer or Plotter (Optional) ..... 1-6
SETTING UP FOR GPIB OPERATION ..... 1-6
Connecting the Equipment ..... 1-7
Configuring the 2711 or 2712 ..... 1-8
Placing the 2711 or 2712 Online ..... 1-8
Setting the GPIB Device Address ..... 1-8
The Power-on SRQ ..... 1-9
Setting the Message Terminator ..... 1-9
Setting the TALK ONLY Option ..... 1-10
Configuring the Device Driver ..... 1-10
Installing the Device Driver ..... 1-12
Configuring the (Optional) Printer or Plotter ..... 1-12
Communicating with the 2711 and 2712 ..... 1-13
Preparing the Software ..... 1-14
A GPIB Instrument Control Program ..... 1-15
RS-232 OPERATION ..... 1-19
OPERATION OVER THE RS-232 INTERFACE ..... 1-20
System Controller ..... 1-21
Software Device Driver ..... 1-22
2711 and 2712 Equipped With the RS-232 Option ..... 1-22
Interconnecting Cable ..... 1-22
Application Software ..... 1-22
Printer or Plotter (Optional) ..... 1-22
SETTING UP FOR RS-232 OPERATION ..... 1-23





GPIB SIGNAL LINE DEFINITIONS ..... A-11
Transfer Bus (Handshake) ..... A-12
Management Bus ..... A-14
IFC (Interface Clear) ..... A-14
ATN (Attention) ..... A-14
SRQ (Service Request) ..... A-14
REN (Remote Enable) ..... A-14
EOI (End Or Identify) ..... A-15
INTERFACE FUNCTIONS AND MESSAGES ..... A-15
RL (Remote-Local Function) ..... A-16
T/TE and L/LE (Talker and Listener Functions) ..... A-16
SH and AH (Source and Acceptor Handshake Functions) ..... A-17
DC (Device Clear Function) ..... A-18
DT (Device Trigger Function) ..... A-18
C, SR, and PP (Controller, Service Request, Parallel Poll Functions) ..... A-19
Taking Control (Asynchronous or Synchronous) ..... A-19
Performing a Serial Poll ..... A-20
Performing a Parallel Poll ..... A-20

**Appendix B — RS-232 Concepts**
INTRODUCTION TO RS-232 COMMUNICATIONS ..... B-1
RS-232 Signal Components ..... B-2
IMPLEMENTATION OF THE RS-232 INTERFACE ..... B-3
RELATED DOCUMENTATION ..... B-7

# OPERATORS SAFETY SUMMARY

The general safety information in this part of the manual is for operating personnel. Specific warnings and cautions may also be found throughout the manual where they apply.

## TERMS IN THIS MANUAL

This CAUTION symbol identifies conditions or practices that could result in damage to the equipment or other property.

This WARNING symbol identifies conditions or practices that could result in personal injury or loss of life.

## TERMS MARKED ON EQUIPMENT

**CAUTION** indicates a personal injury hazard not immediately accessible as one reads the markings, or a hazard to property, including the equipment itself.

**DANGER** indicates a personal injury hazard immediately accessible as one reads the marking.

⚠ means "Caution, refer to the manual for additional information."





## AC POWER SOURCE

CAUTION

*To prevent damage to the Spectrum Analyzer, operate it only from appropriate AC mains sources.*

Damage to the instrument can occur if the 50-60 Hertz AC power source applies more than 250 VAC rms between conductors, or between either conductor and ground. See Section 2, ***Specifications***, in the ***2711 Spectrum Analyzer User*** manual or the ***2712 Spectrum Analyzer User*** manual for additional information.

## DC POWER SOURCE

The 2711 and 2712 Spectrum Analyzers can be powered from the optional model 2704/2705 DC-to-AC Inverter and external DC battery pack. These spectrum analyzers will run for about one hour on a fully charged 2705 Battery Pack, and for extended periods of time on alternate DC sources. See the ***2704 DC-to-AC Inverter and 2705 Battery Pack*** instruction manual for further information.

## PRODUCT GROUNDING

CAUTION

*To prevent potentially hazardous voltages from existing on the exposed metal parts of the 2711 or 2712, do not disconnect the spectrum analyzer's protective ground.*

The 2711 and 2712 Spectrum Analyzers are earthed by the protective grounding lead of their AC power cord. Upon loss of the protective ground connection, all accessible conductive parts of the spectrum analyzer can render an electric shock.

## INPUT POWER AND VOLTAGE LIMITATIONS

**CAUTION**

*The safe maximum total RF input power for the Spectrum Analyzer is +20 dBm (+67 dBmV).*

Total input power above the rated maximum can cause damage to the instrument, and voids the factory warranty.

## USE THE PROPER FUSE

For continued fire protection, observe the fuse specifications located on the rear panel of the 2711 and 2712 Spectrum Analyzers.

## GENERAL PRECAUTIONS

**WARNING**

*Using the 2711 and 2712 Spectrum Analyzers in wet/damp conditions or inclement weather.may result in electric shock or damage to the instrument.*

The 2711 and 2712 Spectrum Analyzers may be operated in any position.

Always allow at least 2 inches (5.1 cm) of clearance adjacent to the ventilation holes at the sides, bottom, and back of the spectrum analyzer case.

The spectrum analyzer is intended for portable operation, and can be used outdoors in fair weather.

The 2711 and 2712 Spectrum Analyzers can be damaged by incorrect AC supply voltages, RF inputs that exceed the maximum ratings, operation in very high temperatures or without adequate ventilation, immersion in liquids, and physical abuse.





To avoid explosion, do not operate this product in explosive atmospheres unless it has been specifically certified for such operation.

To avoid personal injury, do not remove covers or panels, or operate the spectrum analyzer without the protective covers installed.

To avoid the possibility of overheating, do not operate the 2711 or 2712 Spectrum Analyzer in a carrying case.

Refer internal service and adjustment to qualified service personnel.

# Section 1 — Introduction

Replace this page with the tab divider of the same name.

# SECTION 1
# INTRODUCTION TO PROGRAMMING

The Tektronix 2711 and 2712 Spectrum Analyzers allow remote control of their functions when equipped with a communications port. The following instrument configurations provide an IEEE Standard 488 General Purpose Interface Bus (GPIB) or an RS-232 data communications interface.

- 2711 Option 03 — GPIB port
- 2711 Option 08 — RS-232 port
- 2712 (Standard) — GPIB port
- 2712 Option 08 — RS-232 port

With a desk top computer and an appropriate control program, you can configure front panel settings (except those intended for local use only, such as INTENSITY) and acquire, transfer, process, and analyze data remotely.

The command set and message structure are almost identical for the GPIB and RS-232 interfaces. However, a few interface-specific considerations, such as communications parameters and protocols, are different. The setup for each interface is described separately in this section.

**NOTE**

*If your instrument is equipped with the GPIB interface then continue with the next subsection,* ***GPIB Operation.*** *Otherwise, turn to the* ***RS-232 Operation*** *subsection and follow the instructions there.*

## GPIB OPERATION

In addition to conformance with the IEEE 488 Standard, the 2711 and 2712 adhere to the Tektronix Interface Standard for GPIB Codes, Formats, Conventions, and Features. This standard promotes ease of operation and, so far as possible, makes this spectrum analyzer compatible with other Tektronix instruments and with GPIB instruments from other manufacturers.





The IEEE 488 Standard establishes electrical levels, connector configuration, and signal protocols for communication between two or more electronic instruments using a common multi-line bus structure. The bus structure, which is known as the GPIB, consists of eight data lines, eight dedicated control signal lines, a shield, and various grounds.

Data are transferred via eight data lines in a bit parallel, byte serial fashion. That is, the eight bits of a data byte are placed on the eight data lines simultaneously. As soon as they are transferred, the next 8-bit data byte is placed on the lines and transferred. Data consist of instrument commands and queries, control settings, parameter values, or display information. The eight control lines are divided into three transfer control (handshake) lines and five interface management lines. Handshaking and interface management are necessary because the bus operates asynchronously, meaning that signals can be generated by one instrument without regard for what another may be doing or the rate at which the instrument can carry out an operation. For instance, two instruments may try to send information simultaneously, or a high speed instrument may try to send data to a low speed instrument.

Instruments connected to the bus are designated as talker, listener, or both talker and listener. A listener can only receive information over the bus and a talker can only send information. A talker and listener can do both (but not simultaneously).

One instrument is usually designated as the system controller. This is generally a computer which determines through software when specific instruments are activated as talkers or listeners. Each instrument is assigned a unique address between 0 and 30, but only 15 instruments can be connected to the bus simultaneously.

The following example illustrates how data transfer typically takes place (except in the case of abnormal events; see ***Status Reporting*** in Section 5).

1. The instrument on the bus that is designated as system controller determines (through operator intervention or program control) that it needs to send a message to one of the other instruments.

2. Using the data and interface management lines, the controller first addresses the desired instrument as a listener. This is called LISTENING an instrument.

3. Normally the instruments on the bus are idle and signal via the handshake lines when they are ready to receive data. The controller then places the first byte of the message on the bus, indicating the type of information it wants (for instance, the current signal amplitude).

4. The controller then signals, also via the handshake lines, that the data byte is ready.

5. As the listener accepts the data byte, it signals over the handshake lines that it has done so. The controller then removes the data from the data lines.

6. The process from steps 3, 4, and 5 repeats until the entire message has been transferred.

7. The controller indicates that the last data byte has been sent. Depending on the option selected, this is done by signaling over the EOI interface management line simultaneously with the last data byte, or by appending the ASCII codes for carriage return (CR) and line feed (LF) to the end of the message and simultaneously signaling EOI.

8. When the message is complete, the controller normally UNLISTENS the instrument. If a message requires a response, the controller then addresses the instrument as a talker (TALKS the instrument).

9. Now the instrument places the first byte of the response on the data bus and signals that it is ready.

10. After the controller reads the byte, it signals (over the handshake lines) that it has done so and is ready to receive more data. The process repeats until EOI is detected, at which point the controller normally UNTALKS the instrument.

This process is transparent to you. It is carried out by the spectrum analyzer, the GPIB board in your controller, and the device driver software (generally supplied with the GPIB board). In the following subsections you will learn how to set up your spectrum analyzer for GPIB operation. See ***Appendix A*** for additional information concerning IEEE 488 and the GPIB.





# OPERATION OVER THE GPIB

You need the following equipment to operate the 2711 and 2712 Spectrum Analyzers over the General Purpose Interface Bus (GPIB).

- System controller
- Software device driver
- 2711 or 2712 equipped with the GPIB interface
- Interconnecting cable
- Application software
- (Optional) Printer or Plotter

Figure 1-1 shows a small system with a printer and plotter.

## System Controller

The system controller can be any general purpose computer equipped with a GPIB board. Specially built controllers can also be used, but are beyond the scope of this manual. The techniques and programs discussed in this manual are appropriate to the IBM family of personal computers (PCs) and their function-alike counterparts, which support the MS-DOS, PC-DOS, or OS/2 environments.

To function as a controller, your computer must be equipped with a GPIB board. Tektronix supplies three National Instruments GPIB boards for your convenience:

- PC-GPIB Package provides a PCII/IIA board; order S3FG210
- AT-GPIB Package provides a 16-bit AT Bus interface board; order S3FG220
- MC-GPIB Package provides a 16-bit Micro Channel interface board; order S3FG230

## Software Device Driver

The device driver is a program (usually supplied with the GPIB board) that tells your computer how to access the board. For the National Instruments PCII, PCIIA, or PCII/IIA GPIB boards, the device driver is a file named `GPIB.COM`. An additional program is usually supplied that enables you to correctly configure the driver by providing information such as the instrument address and the type of message terminator. The National Instruments program is named `IBCONF.EXE`.

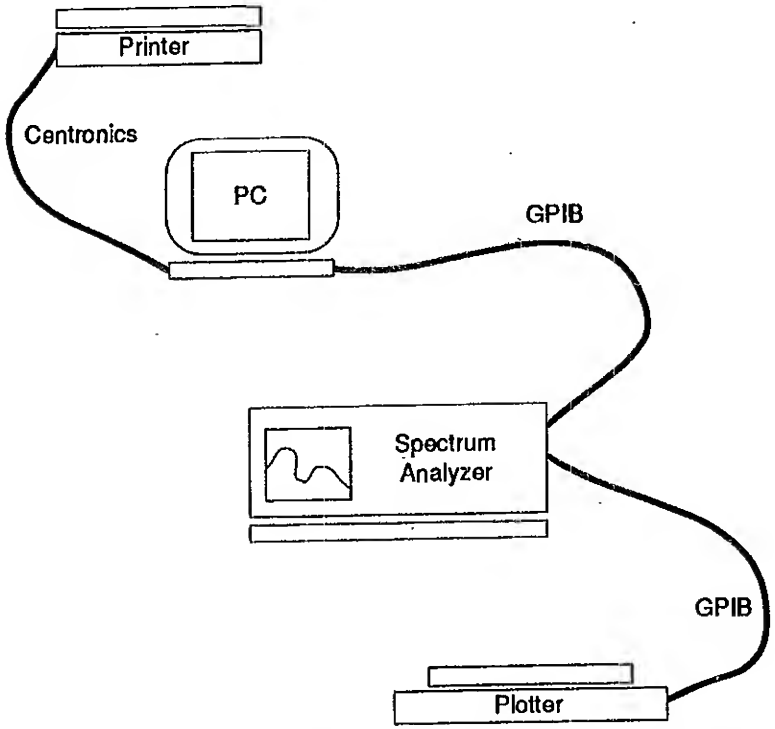



**Figure 1-1. Typical Small Instrument System for GPIB.**

### 2711 and 2712 Equipped with the GPIB

Your 2711 or 2712 must be equipped with the GPIB interface to operate over the General Purpose Interface Bus. Proceed to ***RS-232 Operation*** later in this section for configuration information if your instrument has an RS-232 interface. Press [UTIL] [4] [9] to see a list of the installed options and capabilities.

### Interconnecting Cable

An appropriate interconnect cable is required to connect the controller to the spectrum analyzer. The cable is supplied as part of the Tektronix GURU II package, or it may be purchased separately from Tektronix by ordering P/N 012-0991-01 (1 meter) or 012-0630-01 (2 meter).





### Application Software

Application software is the program or programs which control and acquire data from the spectrum analyzer. You can construct your own programs with the information in this manual. However, you will need the applications interface software supplied by the GPIB board manufacturer. For the PCII/PCIIA board and the QuickBASIC language, these programs have names such as `QBIB4.OBJ`, `QBIB4728.OBJ`, and `QBDECL4.BAS`. The programs include the BASIC device function calls which enable you to communicate easily over the GPIB. The function calls are an integral part of your application programs.

### Printer or Plotter (Optional)

A printer, a plotter, or both can be added to your system to provide hard-copy output. Printers are preferred for character-based data such as parameter values or instrument settings. Plotters provide superior results when displaying graphical data. A convenient approach is to install a printer on a parallel port of the controller and a GPIB-compatible plotter on the bus. With this approach, you can plot graphical data directly from the spectrum analyzer when the controller is not available. See ***Setting the Talk Only Option*** later in this section.

## SETTING UP FOR GPIB OPERATION

Your equipment must be correctly configured before GPIB operations can be performed. The following tasks must be completed.

- Installation of cables between the system components
- Configuration of the the spectrum analyzer and device driver
- Installation of the device driver into controller memory
- Configuring the (optional) printer and/or plotter

This section describes each task in detail.

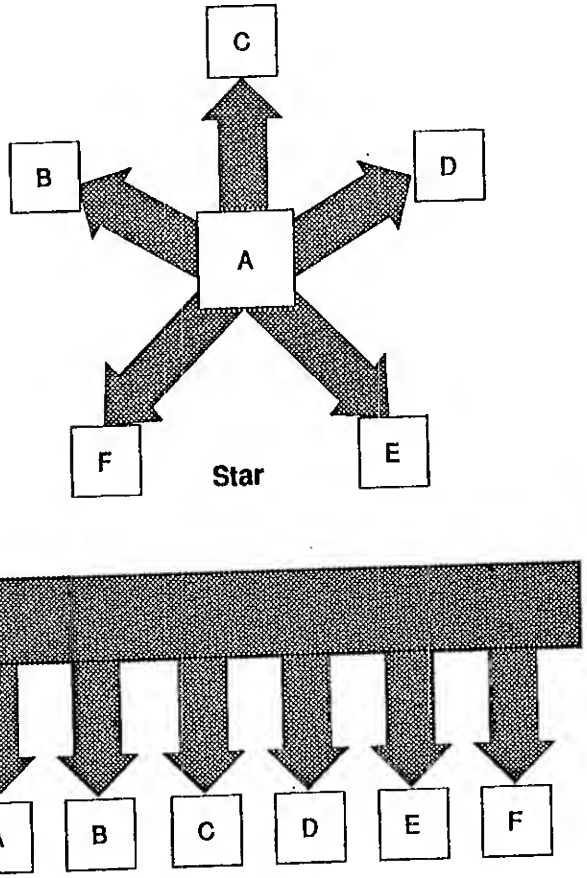



**Figure 1-2. Connecting Multiple Instruments on the GPIB.**

## Connecting the Equipment

If your system consists of a controller and spectrum analyzer, you can simply connect one end of the interconnecting cable to each instrument. A star configuration, daisy chain configuration, or combination of these (Figure 1-2) should be used when more than two instruments are on the bus. Up to 15 instruments can be connected. To maintain electrical





performance of the bus, use only one 2-meter cable per instrument, and ensure that at least 2/3 of the connected instruments are powered up.

## Configuring the 2711 or 2712

Turn on the power to the spectrum analyzer. Then press:

[UTIL][4][0][0]

A GPIB PORT CONFIGURATION Menu appears. It should resemble the one shown in Figure 1-3. You will use this menu to configure the GPIB parameters.

## Placing the 2711 or 2712 Online

Item 0 of the GPIB PORT CONFIGURATION Menu, STATUS, controls the GPIB ONLINE/OFFLINE status. After all preparations have been completed and GPIB operations are ready to begin, press [0] on the KEYPAD to toggle item 0 until the STATUS indicates ONLINE. The spectrum analyzer is then ready to exchange information over the GPIB.

## Setting the GPIB Device Address

Item 1 of the GPIB PORT CONFIGURATION Menu, GPIB ADDRESS, sets the spectrum analyzer's GPIB device address. You must assign a primary address to the spectrum analyzer (the 2711 and 2712 do not support secondary addresses). The address can have a value from 0 through 30. However, addresses 0 and 30 are usually reserved for system controllers. The address you assign is not critical, but it must not be the same address used for any other instrument on the bus.

**NOTE**

*You must assign the same GPIB address to the spectrum analyzer that was used when configuring the device driver for the spectrum analyzer.*

To assign the address, select item 1, GPIB ADDRESS, from the GPIB PORT CONFIGURATION Menu. Follow the on-screen prompts to enter the desired address using the KEYPAD for data entry. If the spectrum analyzer is the only instrument on the bus, we suggest using 1 as the address. The address you set is read immediately by the spectrum analyzer and is permanently retained in non-volatile memory.

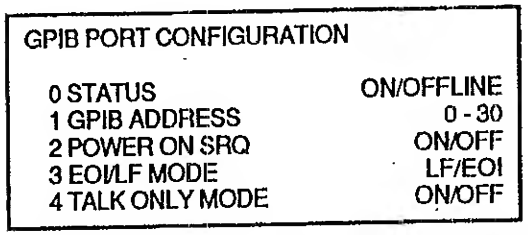



**Figure 1-3. The Spectrum Analyzer's GPIB PORT CONFIGURATION Menu.**

## The Power-on SRQ

Item 2 of the GPIB PORT CONFIGURATION Menu, POWER ON SRQ, causes the spectrum analyzer to produce an SRQ at power up. To generate a POWER ON SRQ, press [2] on the KEYPAD until the status changes to ON.

Normally there is no need to have the spectrum analyzer generate an SRQ when it powers up. Therefore, the default setting of item 2, POWER ON SRQ, is OFF. However, some test sequences require that the power to the spectrum analyzer is removed (power down). Under these conditions you may desire the program to sense the return of power.

## Setting the Message Terminator

Item 3 of the GPIB PORT CONFIGURATION Menu, EOI/LF MODE, selects the message terminator. Whenever a message is transmitted over the bus, the instrument sending the message must signify to other instruments on the bus (including the system controller) that the message has been completed. This is done in one of two ways.

- The interface management line named End Or Identify (EOI) is brought to its low state simultaneously with the last data byte that is transmitted.
- The ASCII codes for carriage return (CR) and line feed (LF) are appended to the message string. EOI is still asserted (brought to its low state) simultaneously with the transmission of LF.

All Tektronix instruments and controllers are equipped to use the EOI selection. You should, therefore, toggle item 3 of the GPIB PORT CONFIGURATION Menu until its status changes to EOI. The LF OR EOI setting is included for controllers which do





not use the EOI signal line. The selection you choose is permanently retained in non-volatile memory.

## Setting the TALK ONLY Option

Item 4 of the GPIB PORT CONFIGURATION Menu, TALK ONLY MODE, selects the spectrum analyzer's TALK ONLY mode.

TALK ONLY mode must be selected to send the spectrum analyzer's output directly to a plotter without the need of a controller. Complete these steps to send the spectrum analyzer's display directly to a plotter.

- Disconnect all instruments except the spectrum analyzer and the plotter from the bus
- Place the plotter in the LISTEN ONLY mode (usually done with controls on the plotter)
- Press [UTIL] [4] [0] and then press [4] until the TALK ONLY status indicates ON
- Press the front panel key labelled [PLOT]

TALK ONLY mode must be disabled when the spectrum analyzer is used with a controller, because the spectrum analyzer must talk to and listen to the controller. To use the spectrum analyzer with a controller, press [4] on the KEYPAD until the status indicates OFF. The system controller will determine when the spectrum analyzer should be addressed as a talker or listener.

## Configuring the Device Driver

Instructions for configuring the device driver should be included with your GPIB board. For example, complete the following steps when using a National Instruments PCII/IIA board.

- Run the `IBCONF.EXE` program to configure the driver
- Follow the on-screen prompts and ensure that the BOARD CHARACTERISTICS screen resembles one of those shown in Figures 1-4 and 1-5
- Create or edit the DEVICE CHARACTERISTICS screen for a device named `TEK_SA` (see Figure 1-6).

**NOTE**

*You must assign the same GPIB address to the spectrum analyzer that was used when configuring the device driver is for the spectrum analyzer. Use the EOI message terminator for all Tektronix controllers.*

| | |
|---|---|
| Primary GPIB Address | 0 |
| Secondary GPIB Address | NONE |
| Timeout setting | T30s |
| EOS byte | 00H |
| Terminate Read on EOS | no |
| Set EOI with EOS on Write | no |
| Type of compare on EOS | 7-bit |
| Set EOI w/last byte of Write | yes |
| GPIB-PC Model | PC2 |
| Board is System Controller | yes |
| Local Lockout on all devices | no |
| Disable Auto Serial Polling | yes |
| High-speed timing | no |
| Interrupt jumper setting | 7 |
| Base I/O Address | 2B8H |
| DMA channel | 1 |
| Internal Clock Freq (in MHz) | 8 |

**Figure 1-4. National Instruments PCII Board Characteristics.**

| | |
|---|---|
| Primary GPIB Address | 0 |
| Secondary GPIB Address | NONE |
| Timeout setting | T30s |
| EOS byte | 00H |
| Terminate Read on EOS | no |
| Set EOI with EOS on Write | no |
| Type of compare on EOS | 7-bit |
| Set EOI w/last byte of Write | yes |
| GPIB-PC Model | PC2A |
| Board is System Controller | yes |
| Local Lockout on all devices | no |
| Disable Auto Serial Polling | yes |
| High-speed timing | no |
| Interrupt jumper setting | 7 |
| Base I/O Address | 02E1H |
| DMA channel | 1 |
| Internal Clock Freq (in MHz) | 6 |

**Figure 1-5. National Instruments PCIIA Board Characteristics.**





| | |
|---|---|
| Primary GPIB Address | 1 |
| Secondary GPIB Address | NONE |
| Timeout setting | T30s |
| EOS byte | 00H |
| Terminate Read on EOS | no |
| Set EOI with EOS on Write | no |

**Figure 1-6. TEK_SA Device Characteristics.**

## Installing the Device Driver

Before your computer can transfer information over the GPIB, it must know how to access the GPIB board and the spectrum analyzer. The device driver tells it how. If you are using a National Instruments PCII/IIA board, the device driver is a program named `GPIB.COM` created and modified by another National Instruments program named `IBCONF.EXE`. If you are using a board from another manufacturer, the appropriate driver should have accompanied your board.

The device driver program must be installed whenever you wish to use the GPIB. Use the following procedure. Refer to your DOS manual if you need help creating or modifying files.

- Copy `GPIB.COM` to your computer's root directory.
- Add the following line to your `CONFIG.SYS` file.

  ```
  device=GPIB.COM
  ```

  If `CONFIG.SYS` does not already exist in the controller's root directory, create this file.
- Reboot your controller.

The GPIB device driver is loaded into memory whenever you boot your computer. It then remains in memory until the computer is turned off or until a warm boot is performed.

## Configuring the (Optional) Printer or Plotter

A variety of printers and plotters are available that can be used with your system. We recommend a serial or parallel printer connected to the appropriate computer port, and/or an HPGL-compatible plotter connected to the GPIB. This arrangement enables you to send data directly from the spectrum analyzer to the plotter when the system controller is unavailable. The

Tektronix HC100 plotter is recommended. Its four pens provide a useful complement to the four-trace capability of the 2711 and 2712.

### Printer Configuration

The configuration of the printer is independent of the GPIB. Consult your printer and computer manuals for information about setting up the printer and corresponding computer communications port.

### Plotter Configuration

Plotter configuration procedures vary. Consult your plotter manual for the configuration appropriate to your plotter.

When using a Tektronix HC100 plotter, set its rear-panel DIP switches as follows:

| | | | | 16 | 8 | 4 | 2 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| OFF / ON | GPIB | HPGL | STD | Address (Listen Only = 31) | | | | | 0 |
| | Down | Down | Down | Up | Up | Up | Up | Up | |

All bits should be set when the Tektronix HC100 plotter is in LISTEN ONLY mode, and its power must be cycled to load the settings into memory. You must also correctly configure the plotter DEVICE CHARACTERISTICS using the `IBCONF` file.

**NOTE**

*Be sure to use the same GPIB address for the HC100 DIP switches and the DEVICE CHARACTERISTICS.*

## Communicating With the 2711 and 2712

The GPIB enables remote or automated control of instruments on the bus, in this case, a spectrum analyzer. An application program (often called a test, measurement, or control program) determines spectrum analyzer operations by exchanging messages with the spectrum analyzer. The messages can be of the generic GPIB type, or they can be instrument-specific.

Generic messages are usually carried out by GPIB hardware and GPIB device driver without intervention by the operator or programmer. They typically implement routine housekeeping chores such as instrument addressing, handshaking, requesting service, or terminating messages.





The instrument-specific messages are also referred to as device-dependent messages. They are generally understood by, and meaningful to, only the instrument or class of instruments for which they are designed. The organization of the instrument-specific messages is explained in the next section of this manual. Section 3, ***Functional Groups***, provides a summary of the messages. Section 4, ***Command and Query Definitions***, describes the individual messages in detail, and Section 6, ***Programming***, provides some programming examples for the National Instruments GPIB/2711 or 2712 combination working in the QuickBASIC environment.

The spectrum analyzer is addressed as a talker or listener to send or receive messages, depending on whether messages are being sent to or received from the system controller. The GPIB system software provided with your GPIB card automatically addresses the spectrum analyzer as a talker or listener depending on the callable subroutine used. The device-dependent messages are then transferred between the controller and the spectrum analyzer over the GPIB as one or more eight-bit bytes of information. Proficiency in controlling the spectrum analyzer is the key to programming these messages efficiently.

## Preparing the Software

After completing the set up procedures your equipment is ready for GPIB operation, but you must still provide the software needed to control the spectrum analyzer. When creating new software this is usually a two step process. The first step is to establish the programming environment. Next you can create and run the control program. If you are using ready-made control software, simply follow the supplier's instructions.

When creating your own QuickBASIC software, you must ensure that QuickBASIC has the necessary GPIB information. Use the following procedure. Refer to your DOS manual if you need help creating or modifying files.

- Copy the files `QBIB4.OBJ`, `GPIB.QLB`, `GPIB.LIB`, `BQLB45.LIB`, and `QBIB4728.OBJ` from the National Instrument disk to the QuickBASIC directory.
- Create the Quick library by typing this command from the DOS command line:

```
LINK /Q QBIB4.OBJ QBIB4728.OBJ,
GPIB.QLB,,BQLB45.LIB
```

- To make your QuickBASIC program a stand-alone *.EXE file, you need an additional library file. Type this command from the DOS command line:

```
LIB GPIB.LIB + QBIB4.OBJ + QBIB4728.OBJ;
```

- Start QuickBASIC using this command:

```
QB /L GPIB.QLB
```

This procedure ensures that the National Instruments GPIB subroutines needed to control devices on the bus are present in the QuickBASIC environment. When using another version of QuickBASIC, use the analogous files and procedures indicated in the READ-QB.DOC document file from National Instruments.

## A GPIB Instrument Control Program

You will learn more about controlling the spectrum analyzer in Sections 4 through 6 in this manual. However, we have provided a simple program here so you can check the operation of your system and observe typical interactions between the controller and spectrum analyzer. Be aware that the program is very basic; it contains no error checking and may hang up the controller (requiring you to reboot) if incorrect or unacceptable commands or queries are entered. It will, however, accept upper- or lower-case entries.

Alternately, you can use the IBIC program supplied with the National Instruments PCII/IIA GPIB board. It enables you to communicate with the spectrum analyzer, but requires that you learn how to use a few simple subroutines such as IBWRT() and IBRD(). See your National Instruments documentation for details.

Follow these steps to use the example program located on the following pages (REM $INCLUDE: 'QBDECL4.BAS').

1. Start QuickBASIC according to the instructions in the preceding subsection. Enter the program. Be sure to enter the program exactly as it is written. The spectrum analyzer must be named TEK_SA by the IBCONF program.

2. Place the spectrum analyzer ONLINE by selecting item 0 from the GPIB PORT CONFIGURATION Menu (press [UTIL] [4] [0] [0]). The instrument is nominally ONLINE, but is not yet handshaking with the controller.





3. Start the program. The computer display shows:

```
2711 (or 2712) SHOULD NOW BE HANDSHAKING
NDAC SHOULD BE DISPLAYED
PRESS ANY KEY TO CONTINUE
```

When the spectrum analyzer is handshaking with the controller, NDAC (Not Data ACcepted) is displayed at the lower right of the spectrum analyzer's screen. NDAC is asserted most of the time. It is unasserted only briefly following receipt of a message to indicate that the message has been accepted (see ***Appendix A***).

4. Press any key. The word REMOTE should appear at the lower left of the spectrum analyzer's screen and the controller should display these messages:

```
2711 (or 2712)  SHOULD NOW BE IN REMOTE MODE
PRESS ANY KEY TO CONTINUE
```

The National Instruments software places the spectrum analyzer in remote mode whenever a message is sent (the message HDR ON was transmitted). Unless the GPIB local lockout command is issued, any spectrum analyzer key which alters its measurement status may be pressed to return the spectrum analyzer to local mode. (For example, the [MENU] keys do not change the status, but items selected from a menu may change the status.)

5. Press [VID FLTR] twice. The REMOTE message should disappear from the spectrum analyzer screen.

6. Press any key. This message should appear.

```
ENTER MESSAGE TO SEND
```

Enter the message you want to send, which can be either a command or query. For example, enter this query requesting the spectrum analyzer to identify itself:

```
ID?
```

7. REMOTE should reappear on the spectrum analyzer screen. The words TALKER or LISTENER will also appear momentarily. They are displayed when the spectrum analyzer enters the indicated mode, but because of timing considerations, they do not always appear for short commands, queries, and responses. You should see a response similar to the following one on the controller's screen.

The reply will look like this one:

```
ID TEK/2711 (or 2712),V81.1,"VERSION 10.11.91 FIRM
WARE","VIDEO MONITOR", "GPIB","COUNTER","NVM 12.88";
```

The actual response depends on the options in your instrument.

8. You will then be asked

```
SEND MORE (Y/N)?
```

Enter "Y" to send more; other answers end the program.

```
REM $INCLUDE: 'QBDECL4.BAS'
  RD$ = SPACE$(3000)
  CLS
  CALL IBFIND("GPIB0", BD%)
  V% = 0
  CALL IBSRE(BD%, V%)
  CALL IBFIND("TEK_SA", BD%)
  PRINT "2711 (or 2712) SHOULD NOW BE HANDSHAKING"
  PRINT "NDAC SHOULD BE DISPLAYED"
  PRINT:PRINT "PRESS ANY KEY TO CONTINUE"
  DO WHILE INKEY$ = ""
     LOOP
  WRT$ = "HDR ON"
     CALL IBWRT(BD%, WRT$)
  PRINT "2711 (or 2712) SHOULD NOW BE IN REMOTE
  MODE"
  PRINT:PRINT "PRESS ANY KEY TO CONTINUE"
  DO WHILE INKEY$ = ""
     LOOP
SEND.RCV:
  CLS
  PRINT : PRINT "ENTER MESSAGE TO SEND"
     PRINT : INPUT WCALL IBWRT(BD%, WRT$)
  QUES = INSTR(1, WRT$, "?")
  HOLD.TIME = TIMER
  DO WHILE TIMER < HOLD.TIME + 1
     LOOP
  IF QUES = 0 THEN GOTO MORE
     CALL IBRD(BD%, RD$)
```





```
  PRINT : PRINT "THE REPLY IS:"
  PRINT : PRINT MID$(RD$, 1, IBCNT%)
MORE:
  PRINT : PRINT
  INPUT "SEND MORE (Y/N)? "; Y$
  IF Y$ = "Y" THEN GOTO SEND.RCV
END
```

## RS-232 OPERATION

If your spectrum analyzer is equipped with a GPIB instrument bus, you can skip this subsection.

The 2711 and 2712, when equipped with the RS-232 interface, follows the EIA Standard RS-232-D. EIA Standard RS-232-D revises RS-232-C so it conforms with international standards CCITT V.24, V.28 and ISO IS2110. This standard establishes electrical levels, connector configuration, and signal protocols for communication between two devices called the DCE (data circuit-terminating equipment) and the DTE (data terminal equipment). The 2711 or 2712 implements the DTE end of the interface.

Note that the RS-232 interface is *NOT* a bus. Only one device can be connected to the instrument's RS-232 interface. Unlike a GPIB interface, RS-232 does not support device addresses or serial polling.

For example, if a computer is connected to the spectrum analyzer's RS-232 interface, a printer or plotter could not be connected to the spectrum analyzer without first disconnecting the computer. To plot screen data directly from the spectrum analyzer, you would first have to disconnect the computer and then connect your printer or plotter.

The 2711 and 2712 RS-232 interface requires a minimum of three lines for operation.

- Transmit data (TXD)
- Receive data (RXD)
- Ground (GND)

If hardware handshake is required, two additional lines must be supplied in the cable.

- Clear to Send (CTS)
- Request to Send (RTS)

The section entitled ***Selecting a Data Flow Control Method***, located later in this section, describes the use of these lines for hardware flow control.





EIA Standard RS-232-D defines other lines typically used for modem control and handshaking. The 2711 and 2712 can operate using the minimum wiring configuration. If the appropriate handshake lines are provided, a printer or plotter that expects handshaking over the RS-232 interface may be used. ***Appendix B*** contains additional cabling information.

Data bits are transferred serially, one bit at a time, over the interface. Data consists of instrument commands and queries, control settings, parameter values, or display information.

If a computer is connected to the spectrum analyzer via the RS-232 interface, the computer's serial interface (called a COM port if the controller is an MS-DOS) must be correctly configured beforehand. Programmed commands and data can then be transmitted over the Interface to the instrument.

If a query such as `FREQ?` is transmitted, the spectrum analyzer formats its response immediately and sends it back to the computer. The control program must be ready to receive the incoming data. In the following subsections you will learn how to set up your 2711 or 2712 for RS-232 operation. ***Appendix B*** provides additional information concerning RS-232 implementation for the 2711 and 2712 (including wiring for connectors and null-modem adapters).

## OPERATION OVER THE RS-232 INTERFACE

The following items are needed to operate the 2711 and 2712 Spectrum Analyzers over the RS-232 interface.

- System controller or terminal
- Software device driver
- 2711 or 2712 equipped with an RS-232 interface
- Interconnecting cable
- Application software
- (Printer or Plotter Optional)

Figure 1-7 shows two RS-232 system configurations. The top illustration shows a computer (PC) controlling the spectrum analyzer via the RS-232 interface, with a plotter connected to the computer over a Centronics interface. The lower illustration shows the spectrum analyzer connected directly to a plotter via the RS-232 interface.

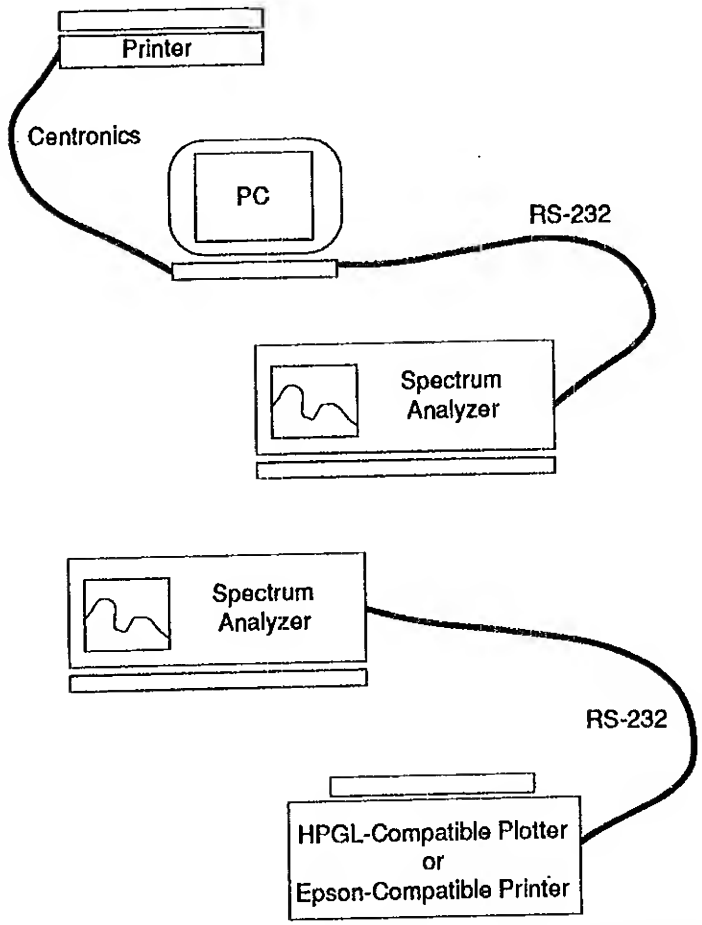



**Figure 1-7. Two RS-232 System Configurations.**

## System Controller

The system controller can be any general purpose computer or terminal equipped with an RS-232 interface (also called a COM port or serial interface). Specially built controllers can be used, but are beyond the scope of this manual. The techniques and programs discussed in this manual are appropriate to the IBM PC family of computers and their function-alike counterparts, which support the MS-DOS, PC-DOS, or OS/2 environments.





## Software Device Driver

The device driver is a program that handles input and output to the RS-232 interface on your computer. The driver for your system depends on the operating system and the programming language you are using. For example, if you are operating a PC, the RS-232 driver configuration may be set with the MS-DOS MODE command. If your control program is written in the BASIC or QuickBASIC language, optional arguments in the OPEN statement can supply RS-232 configuration settings.

## 2711 and 2712 Equipped With the RS-232 Option

Your 2711 or 2712 must be equipped with the RS-232 port. If your instrument is equipped with a GPIB interface, refer to ***GPIB Operation*** earlier in this section. Press [UTIL] [4] [9] to see a list of the installed options and capabilities.

## Interconnecting Cable

An appropriate cable is required to connect between the controller and the spectrum analyzer. The pinout and connector type are identical to the 9-pin connector used for PC/AT type RS-232 interfaces. Such cables are available in most computer stores. For some RS-232 devices, null-modem adapters will be needed. Refer to ***Appendix B*** for further information on connectors and adapters.

## Application Software

Application software is the program or programs that control and acquire data from the spectrum analyzer. You can construct your own programs using the information in this manual. Off-the-shelf software is also available.

## Printer or Plotter (Optional)

A printer or plotter (not both simultaneously) can be connected to the RS-232 interface to provide hard-copy output. A printer is the preferred instrument for character-based data such as parameter values or instrument settings. Plotters provide superior results when displaying graphical data.

A printer or plotter cannot be connected to the spectrum analyzer's interface when a computer is connected. For this reason you must choose between computer control or hard-copy output when working directly from the spectrum analyzer's RS-232 interface. An alternate approach connects the computer to the spectrum analyzer interface while using a

control program to acquire data from the spectrum analyzer. A second RS-232 port, a GPIB port, or a Centronics port on the computer is then used to produce output on a printer or plotter.

## SETTING UP FOR RS-232 OPERATION

Your equipment must be correctly configured before performing RS-232 operations. The following tasks must be completed.

- Installation of cables between the system components
- Configuration of the spectrum analyzer and device driver
- Installation of the device driver into controller memory
- Configuring the (optional) printer or plotter

This section describes each task in detail.

### Connecting the Equipment

Only one device (computer, plotter, or printer) can be attached to the spectrum analyzer's RS-232 interface. For systems consisting of a controller and the spectrum analyzer, you can simply connect one end of the interconnecting cable to each device. Figure 1-7 shows two possible configurations. See ***Appendix B*** for the cable configuration appropriate for your setting.

### Configuring the Spectrum Analyzer

Both devices (the computer and spectrum analyzer) in an RS-232 system must be configured the same way. Before setting up the spectrum analyzer, be sure to check the configuration settings for the device with which you expect to communicate.

To set the spectrum analyzer configuration settings, turn on the power to the 2711 or 2712 and press

[UTIL][4][0][2]

on the KEYPAD. An RS-232 PORT CONFIGURATION Menu appears that is similar to the one shown in Figure 1-8. It enables you to configure the spectrum analyzer's RS-232 parameters.





RS-232 PORT CONFIGURATION

| | |
|---|---|
| 0 STATUS | ONLINE/OFFLINE |
| 1 BAUD RATE | 110 - 9600 |
| 2 DATA BITS | 7/8 |
| 3 PARITY | NONE/ODD/EVEN |
| 4 EOL | CR/LF/CR LF |
| 5 FLOW CONTROL | HARD/SOFT/NONE |
| 6 ECHO | ON/OFF |
| 7 VERBOSE | ON/OFF |

**Figure 1-8. The RS-232 Port Configuration Menu.**

#### Placing the 2711 and 2712 Online

Item 0 of the RS-232 PORT CONFIGURATION Menu, STATUS, controls the RS-232 online/offline status. When the status is set to OFFLINE, the RS-232 interface is ignored; data are neither received nor transmitted. After all preparations have been completed and RS-232 operations are ready to begin, press [0] on the KEYPAD to toggle item 0 until the STATUS indicates ONLINE. The spectrum analyzer is then ready to exchange information over the RS-232 interface.

#### Setting the Baud Rate

Item 1 of the RS-232 PORT CONFIGURATION Menu, BAUD RATE, sets the baud rate of the spectrum analyzer. Baud rate represents how fast data are transmitted across the interface. To select a baud rate, repeatedly press [1] on the KEYPAD until the baud rate you desire is displayed. Baud rates ranging between 110 and 9600 are available.

The number of stop bits used is automatically selected by the spectrum analyzer when you change baud rates. If the baud rate is 110, then two stop bits are selected. One stop bit is selected for all other baud rates.

#### Setting the Number of Data Bits

Item 2 of the RS-232 PORT CONFIGURATION Menu, DATA BITS, selects the number of data bits sent per character. This is either seven or eight. Eight bits must be selected for binary transfers. Press [2] on the KEYPAD to choose between seven or eight data bits.

### Setting Parity

Item 3 of the RS-232 PORT CONFIGURATION Menu, PARITY, determines whether odd or even parity is used for data checking, or it selects no parity checking. Default is NONE. To change the PARITY selection, repeatedly press [3] on the KEYPAD until ODD, EVEN, or NONE is displayed.

### Setting the Message Terminator

Item 4 of the RS-232 PORT CONFIGURATION Menu, EOL, selects the EOL (end-of-line) indicator used to terminate messages sent over the spectrum analyzer's RS-232 interface. The terminator can be CR (carriage return, ASCII 13), LF (line feed, ASCII 10), or CR LF (carriage return followed by line feed). To change the EOL status selection, repeatedly press [4] on the KEYPAD until CR, LF, or CRLF is displayed.

When a controller sends data, the spectrum analyzer interprets either CR or LF as a terminator, independent of the setting.

### Selecting a Data Flow Control Method

Item 5 of the RS-232 PORT CONFIGURATION Menu, FLOW CONTROL, selects between three flow control methods: SOFT, HARD or NONE. An explanation of each follows.

**SOFT:** When the spectrum analyzer sends data through the interface and soft flow control is enabled, CTRL-S (ASCII 19, same as pressing [CTRL] and [S] simultaneously) halts the data stream until CTRL-Q (ASCII 17) is received. Any other character received in the interim is ignored. This type of flow control can be used with a 3-wire setup because additional handshake lines are not needed.

When SOFT control is selected, the spectrum analyzer sends CTRL-S when its input data buffer is within 200 characters of being full. It sends CTRL-Q when the buffer empties to the point at which additional characters can be safely accepted (less than 200 characters remain in the buffer). If the input buffer is allowed to overflow, the spectrum analyzer discards the incoming data and signals an error (Event 372).

**HARD:** When HARD flow control is selected, the instrument sends data as long as the CTS (Clear-To-Send) line is TRUE and halts the data stream if CTS goes FALSE. Additional handshake lines (more than a 3-wire RS-232 implementation) are required to support HARD flow control.





When receiving data and HARD flow control is selected, the spectrum analyzer asserts RTS (Request-To-Send) TRUE until the input buffer is within 200 characters of being full. It then sets RTS FALSE. As with SOFT flow control,data is received while RTS is FALSE, but if the buffer is allowed to overflow the spectrum analyzer signals an error (Event 372) and incoming data is discarded.

**NONE:** No flow control is used.

In general you should follow these rules when selecting a flow control method:

- Do not use SOFT flow control when transmitting file or waveform data (binary transfers) because you cannot guarantee that the ASCII-decimal values corresponding to CTRL-S and CTRL-Q do not appear in the input stream. For files and waveform data specify HARD flow control or NONE.
- If NONE is specified, you must ensure that buffers do not overflow. This can be done by allocating enough buffer space to handle most contingencies. A buffer size of 1200 is sufficient for most purposes. The 2711 and 2712 use a 1200-byte, internal input buffer.

### Selecting the Echo Feature

Item 6 of the RS-232 PORT CONFIGURATION Menu, ECHO, chooses ECHO modes of ON or OFF. Echo mode is intended primarily as a means of interacting with the 2711 and 2712 from a "dumb" terminal, or for testing purposes. Press [6] on the KEYPAD to choose between ON or OFF.

When ECHO is OFF, the spectrum analyzer does not return the characters it receives to the controller. For most cases, ECHO should be OFF. However, set ECHO to ON when using a "dumb" terminal to control the spectrum analyzer.

When ECHO is ON, the spectrum analyzer echoes each character it receives back to the controller. This can cause problems for the control program if it is not expecting the characters. Additional time is required to process each returned character, so it is possible to experience buffer overrun at 9600 baud if the character rate is too high. After each command or query is completed, the spectrum analyzer prompts for further input by returning the string "> " to the controller.

For example, if ECHO mode is ON, ">" appears on the terminal or computer display screen. If the command "VPO?" is entered, the spectrum analyzer returns "VPO?" followed by a normal response to the query, such as "VPOLARITY POSITIVE". It then appends the ">" indicating it is ready to receive additional commands.

Because ECHO mode lets you see each character received by the spectrum analyzer, it is sometimes useful for interactive testing. Following are some important ECHO mode characteristics to keep in mind.

- If SOFT flow control is enabled, CTRL-S or CTRL-Q are not echoed, but they perform their normal functions
- If either CR or LF is received by the spectrum analyzer, it is echoed as the currently selected output terminator
- Any other control character echoes as an up arrow (^) followed by a capital letter; for example, ^X represents pressing [CTRL] and [X] at the same time
- ECHO should not be ON with binary transfers
- If ECHO is ON the prompt character appears on the display under other conditions; 1) when the instrument is powered up or placed on-line with ECHO mode ON, 2) when ECHO is turned on, and 3) after a device clear (break) is received

### Verbose Mode and Error Handling

Item 7 of the RS-232 PORT CONFIGURATION Menu, VERBOSE, turns VERBOSE mode ON and OFF. This feature is provided as an alternative to GPIB SRQ mechanism. It is generally used when controlling the spectrum analyzer with a "dumb" terminal. Press [7] on the KEYPAD to choose between ON or OFF.

When ON, VERBOSE mode forces the spectrum analyzer to respond for each command it receives. The response will be one of the following:

- An event code for an abnormal condition
- A response for a successful query (FREQ?)
- The string "OK" for a successful non-query

Refer to Section 5, ***Status Reporting***, for additional information on error handling for RS-232 equipped instruments.





### Installing and Configuring the Device Driver

If you are using special applications software or a custom RS-232 driver, follow the detailed instructions for installing and configuring the device driver included with it. However, for PC type controllers running MS-DOS, the driver is part of the operating system. You can configure a serial communications port with the MODE command by entering a command similar to the following example.

```
MODE COM1:9600,n,8,1
```

This command configures the COM1 interface to run at 9600 baud, no parity, 8 data bits and 1 stop bit.

**NOTE**

*You must use the same set up information for the controller and the spectrum analyzer.*

A program statement, such as OPEN in the BASIC language, is an alternative way to configure the driver. This method of driver configuration is recommended because it sets the driver to a known, and presumably correct, operating state from within the application program and just prior to actual operation. If the MODE command is used, the last settings applied to the interface must be remembered. If these data are not remembered, your program will not work properly.

### Configuring the (Optional) Printer or Plotter

A variety of printers and plotters are available for use with your system. The serial or parallel printer of your choice may be connected to the appropriate computer port. For example, the Centronics- or GPIB-compatible, 4 pen Tektronix HC100 plotter is recommended. Its four pens provide a useful complement to the four-trace capability of the 2711 and 2712.

A serial printer or plotter, such as the Tektronix HC100 (Option 3), can be attached to the spectrum analyzer's RS-232 interface instead of a computer controller. This arrangement enables data transfer directly from the spectrum analyzer to the printer or plotter with a remote PLOT command when the system controller is unavailable. Of course, the spectrum analyzer must be correctly configured using the SCREEN PLOT CONFIGURATION Menu ([UTIL] [4] [1]).

Figure 1-7, located earlier in this section, shows two alternative configurations using a plotter. A printer could be substituted for the plotter in either configuration.

## Communicating with the Spectrum Analyzer

The RS-232 interface enables remote or automated control of the 2711 and 2712 spectrum analyzers. An application program (often called a test, measurement, or control program) determines 2711 or 2712 operations by exchanging spectrum analyzer-specific messages with the instrument.

The spectrum analyzer-specific messages are also referred to as device-dependent messages. They are generally understood by and meaningful to only the instrument or class of instruments for which they are designed. The organization of the spectrum analyzer-specific messages is explained in the next section of this manual. Section 3, ***Functional Groups***, provides a summary of the messages. Section 4, ***Command and Query Definitions***, describes the individual messages in detail, and Section 6, ***Programming***, provides some programming examples.

Programmed commands and data are transmitted over the interface to the instrument as soon as they are delivered to the driver. If the command is a query (FREQ? for example), the spectrum analyzer formats a response immediately and sends it back to the computer. The control program is responsible for handling incoming data in a timely fashion.

## Preparing the Software

After completing the set up procedures your equipment is ready for RS-232 operation, but you must still provide the software needed to control the spectrum analyzer. When creating new software this is usually a two step process. The first step is to establish the programming environment. Next you can create and run the control program. If you are using ready-made control software, simply follow the supplier's instructions.

The programming requirements for RS-232 control are more complex than those for GPIB operation. Section 6, ***Programming***, contains a complete example of an interactive RS-232 control program. This program is functionally similar to the GPIB program example located earlier in this section (see ***A GPIB Instrument Control Program***).

# Section 2 — Message Structure

Replace this page with the tab divider of the same name.

# SECTION 2
# INSTRUMENT-SPECIFIC MESSAGE STRUCTURE

Generic GPIB messages and instrument-specific messages are exchanged between the system controller and the spectrum analyzer over the GPIB. When the RS-232 interface is installed, communications between the system controller and spectrum analyzer are limited to instrument-specific messages.

Generic GPIB messages (GPIB interface required) exercise control over the bus and carry out routine system operations such as instrument addressing, handshaking, requesting service, and terminating messages. GPIB messages may be transmitted over the handshake lines or interface management lines (uni-line messages), or they may be transmitted over the data lines (multi-line messages). The GPIB hardware and software usually sends and receives these messages in a way that is transparent to the system operator or programmer. Refer to ***Appendix A*** for additional information about uni- and multi-line messages.

The instrument-specific messages exchanged over the GPIB or RS-232 interface control the measurement and display functions of the spectrum analyzer. These messages are always transmitted over the data lines (with the exception of the EOI message terminator).

Instrument-specific messages control parameters such as center frequency, span/division, reference level, and resolution bandwidth. It is the system programmer's task to efficiently compile a series of messages in a script designed to implement specific tests and measurements. The script, which is written in a conventional computer language and embodies specific spectrum analyzer commands and queries, is called a control program.





## WHAT IS A MESSAGE?

An instrument-specific message consists of three or more 8-bit bytes of information that are transferred between the spectrum analyzer and the system controller. Each byte represents an ASCII character or binary data. A message may be an input message or an output message, and it may contain one or more message units.

For instance, here is an example of a message from Mary to John.

```
John

dinner - put in oven; washing machine - start; bank -
withdraw how much?; cat - let her in

Bye bye
```

John's response may resemble this one.

```
Mary

$100

Bye bye
```

He could also respond this way.

```
Mary

Withdraw - $100

Bye bye
```

Both messages are structured in a similar manner. Each contains a salutation (`John` and `Mary`). Each message consists of one or more message units. For instance, Mary's message to John has four units; one of these is a query and is identified by a question mark (?). Each message unit begins with a "header" describing what the message is about (`dinner` and `cat`). The header is separated from its object or argument by a dash (-), which is the argument delimiter.

Message units are separated or delimited by semicolons. John's messages to Mary consist of a single response indicating how much money she should withdraw. If John thinks Mary will remember her own question (`withdraw how much?`), he may simply reply "`$100`" as in the first example. However, to relate his response to her question he may answer "`Withdraw - $100`" as in the second example. The latter form is equivalent to receiving a response from the spectrum analyzer when HDR ON

(see Section 4, ***Command and Query Definitions***) is selected. Both messages close with a message terminator "Bye bye".

The instrument-specific messages are constructed in a similar way. The following definitions clarify the structure.

## Input Message

An input message is one or more message units, along with any message unit delimiters and a message terminator, transmitted from the controller to the spectrum analyzer.

## Output Message

An output message is one or more message units, along with any message unit delimiters and a message terminator, transmitted from the spectrum analyzer to the controller.

## Message Unit

A message unit is a single command, query, or response.

## Message Unit Delimiter

A semicolon (;) must be used to delimit or separate message units in a message. Following the last message unit, the use of a delimiter is optional with one exception. The spectrum analyzer always appends a message unit delimiter as the last data byte when it sends a response.

If desired, you may substitute the line feed character for the semicolon in the case of responses (see the MSGDLM command in Section 4). Do not confuse the line feed character with the optional message terminator discussed next.

## Message Terminator (GPIB)

Messages exchanged over the GPIB interface can be terminated in one of two ways.

- The EOI interface management line is brought low (asserted) simultaneously with the last byte of the message
- ASCII codes for carriage return (CR) and line feed (LF) are appended to the end of message and the EOI interface management line is asserted simultaneously with transmission of the LF character





All Tektronix instruments assert the EOI line. The CR-LF terminator option is provided for instruments which do not use the EOI line. If you are using such an instrument, select the CR-LF option (see ***Setting the Message Terminator*** in the ***Setting Up for GPIB Operation*** part of Section 1). Terminator control is handled automatically by the GPIB hardware and software.

## Message Terminator (RS-232)

When a controller sends data to the 2711 and 2712, the spectrum analyzer interprets both CR and LF as a message terminator. Messages sent by the spectrum analyzer over the RS-232 interface can be terminated in the following ways.

- By CR only (carriage return, ASCII 13)
- By LF only (line feed, ASCII 10)
- By CR-LF (carriage return followed by line feed)

See ***Setting the Message Terminator*** in the ***Setting Up for RS-232 Operation*** part of Section 1 for instructions on configuring the message terminator for RS-232 instruments.

## Command

A command generally consists of a command mnemonic or header, header delimiter, argument(s), and argument delimiter. However, some commands have no arguments, or only one argument, and may not require header or argument delimiters.

## Query

A query consists of a query mnemonic or header, a question mark, header delimiter, and argument. However, many queries do not require an argument.

## Response

A response consists of an optional response mnemonic or header, header delimiter, argument(s), and argument delimiter.

## Mnemonic or Header

A header is a short name associated with each command, query, or response (for example, FREQ, REF, MAR). Whenever possible, choose headers that are mnemonic in nature so the name reminds you of the function.

### Header Delimiter

A header delimiter is a space. Command headers, response headers, and the question mark following a query header are delimited or separated from any following arguments by a space. The space is optional if there are no arguments.

### Argument

An argument is the value(s) that a command, query, or response transfers to or from its associated spectrum analyzer setting(s). For instance, in the command

```
FREQ 200 MHZ
```

the value of 200 MHz is transferred to the center frequency setting. Arguments may be numbers (with or without units), characters, strings, or linked with a colon (:). A block of binary data may comprise the argument of some waveform commands and responses.

### Digit

A digit is any of these numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8 or 9.

### Number Argument

Number refers to a decimal number consisting of one or more digits. Three number formats are possible:

| | |
|---|---|
| nr1 | Integer (no decimal point) |
| nr2 | Floating point number (decimal point required) |
| nr3 | Integer or floating point number in scientific notation (2.0E+3 or 2.000E+3, for example, instead of 2000) |

### Units

Engineering units can be appended to certain number arguments. Except in the case of decibels (dBs), only the first letter of the units is used; the remainder is determined by context. For instance, the command

```
FREQ 10 M
```

is the same as

```
FREQ 10 MHZ
```





The M is interpreted as MHz because frequency corresponds to the `FREQ` command. In a similar manner, the letter M would indicate milliseconds when used with the `TIME 10 M` command.

Commands such as `REFlvl` require the entire dB unit (DBMV, DBM, etc.) to avoid confusion. You can place as many spaces as desired between the number and its units.

Note that responses with numerical arguments DO NOT append units to the argument. You must keep track of the units yourself or use a query that specifically responds with the units currently in use (for instance, the `RLUnit?` query).

### Character Argument

A character argument consists of one or more letters usually expressing a word or mnemonic. For instance, ON and OFF are arguments for commands that control spectrum analyzer functions such as `ARES` (auto resolution bandwidth).

### String Argument

String arguments consist of one or more characters, including spaces, enclosed in quotation marks ("). They are used with commands such as `TITLe` to convey messages meant to be displayed or plotted. Double quotation marks ("") must be used when the quotation marks are to be part of the message.

### Link Argument

Link arguments provide a method of passing related argument parameters. A colon character (:) separates linked arguments. For example, the `VRTdsp` command can set both the vertical display mode and its related scale factor (the command `VRT LOG:5` selects logarithmic display of 5 dB/division).

### Binary Block Argument

A binary block argument is a sequence of binary numbers. The sequence is preceded by the ASCII code for percent (%), a two-byte binary integer representing the number of binary numbers, and a checksum byte. Following the sequence is an additional checksum byte which provides an error check of the binary block transfer.

### Argument Delimiter

A comma (,) must be used to delimit or separate multiple arguments in a message unit. It should not be used as the last character in a message.

## MESSAGE BUFFERING (GPIB)

The spectrum analyzer buffers each input message it receives. Message processing begins as soon as messages are received by the spectrum analyzer; it does not wait for the message terminator. Once processing begins, the spectrum analyzer remains busy until it is done executing the commands in its input buffer unless the process is stopped by the DCL (Device Clear) or SDC (Selected Device Clear) GPIB messages.

If an error is detected while transferring a command or query, the remainder of the message (up to the message terminator) is discarded.

Output data are ready following execution of each query and are passed to the spectrum analyzer's output buffer prior to transmission over the bus. The spectrum analyzer begins to transmit an output message after it is addressed as a talker and the data become available. However, the response terminator is not sent until the command terminator is detected in case there are more queries in the input message.

Output continues from the spectrum analyzer until the end of the information in the output buffer is reached, or until it is interrupted by a DCL (Device Clear), UNT (Untalk), or IFC (Interface Clear) GPIB message. If the spectrum analyzer is interrupted before the buffer is cleared, output will resume if the spectrum analyzer is readdressed as a talker.

The buffer is cleared by the DCL or SDC messages. The spectrum analyzer terminates the output according to the selected message terminator (EOI or CR/LF/EOI) unless it is interrupted.





## MESSAGE BUFFERING (RS-232)

The spectrum analyzer buffers each input message it receives. Message processing begins as soon as messages are received by the spectrum analyzer; it does not wait for the message terminator. Once processing begins, the spectrum analyzer remains busy until it is done executing the commands in its input buffer unless the process is stopped by a BREAK. BREAK is sensed by the interface as a null character together with a framing error. If an error is detected while transferring a command or query, the rest of the message (up to the message terminator) is discarded.

Under RS-232 operation, buffering is handled by specifying a flow control method. Refer to ***Setting Up the RS-232*** in Section 1 for instructions for details describing flow control configuration and use.

HARD flow control, the default method, uses the RTS (Request To Send) and CTS (Clear To Send) handshake wires. In this mode the remaining output line, DTR (Data Terminal Ready), is always asserted TRUE and input lines DCD (Data Carrier Detect) and DSR (Data Set Ready) are ignored.

SOFT flow control uses the CTRL-Q/CTRL-S method. Output lines RTS and DTR are always asserted TRUE and input lines DCD, CTS, and DSR are ignored. This method should not be specified for binary transfers such as waveforms or files.

NONE (No flow control) may also be specified. In this case the user is responsible to ensure that I/O buffers do not overflow. Note that both SOFT flow control or NONE (no flow control) allow the use of a 3-wire interface (GND, TXD, and RXD). In each case RTS and DTR are asserted TRUE and the CTS, DCD, and DSR inputs are ignored. Refer to ***Appendix B*** for more details on RS-232 connector wiring and the implementation of the interface standard by the 2711 and 2712.

# MESSAGE FORMAT

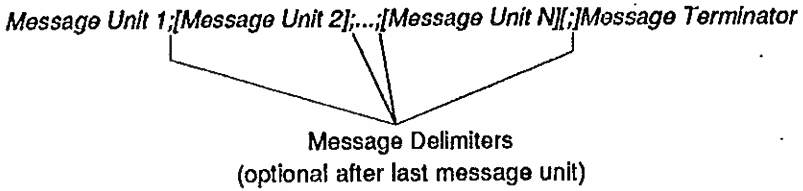
Messages are formatted along structural lines. Each message consists of one or more message units separated by semicolons (;).

**NOTE**

*Line feeds can be selected as message unit delimiters in GPIB responses — see the* MSGDLM *command.*

## Message

Each input or output message can be represented graphically as shown in this example.




Items enclosed in square brackets [] denote optional quantities. Each message unit consists of an individual command, query, or response with all necessary arguments and delimiters. Simple messages may consist of individual commands or queries as in these examples.

```
FREQ 200MHZ

SAVE A:ON

CURVE?
```

However, message units may be combined to form more complex messages like this one.

```
FREQ 200 MHZ;SAVE A:ON;CURVE?
```

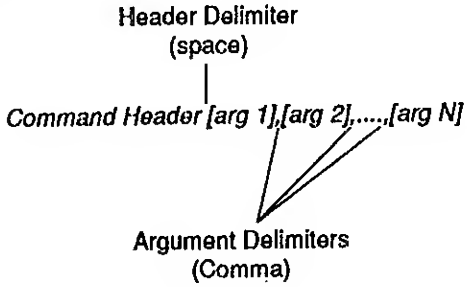
Notice that commands and queries can be mixed in a single message.





## Commands

A command can be represented graphically as shown in this example.




Although multiple arguments are shown, most commands have only a single argument. Following are several examples of specific commands.

```
FREQ 5.5E+6

REFLVL -12 DBMV

VRTDSP LIN:100MV

RESBW 1.0M

SAVE A:ON,B:ON,C:OFF

TIME 5 US
```

These examples illustrate several important characteristics of command formatting.

1. A header delimiter (space) following the header is required.

2. Variable number formatting options are available; arguments are expressed as integers (–12 and 100), floating point numbers (1.0 and .035), and in scientific notation (5.5E+6).

3. The absence of units indicates that the appropriate units are inferred from the command header. For instance, `TIME` implies seconds and `FREQ` or `RESBW` imply Hz. Thus, 5.5E+6 in the `FREQ` command implies 5.5 x $10^6$ Hz or 5.5 MHz.

4. A space between an argument and its units is optional.

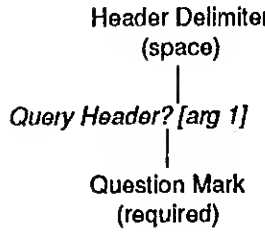
5. Shortened forms of units may be used (M instead of MHZ in the `RESBW 1.0M` command). Only the first letter of the unit is read. The value and the units represented by the letter are dependent on the command it is used with.

For instance, M is interpreted as $10^6$ Hz (MHZ) when used as above, but represents $10^{-3}$ seconds (MSEC) when used with the TIME command. Note however, that commands such as REFLVL require the entire unit to avoid confusion between the various dB units.

6. Linked arguments (VRTDSP) are always delimited by colons. Multiple linked arguments (SAVE) are always delimited by commas.

## Queries

A query can be represented graphically as shown in this example.




Most queries recognized by the 2711 and 2712 have no arguments, but a few have one argument. There are no queries with multiple arguments. Following are several examples of specific queries.

```
FREQ?

REFLVL?

VRTDSP?

RESBW?

VIEW? A

TIME?
```

These examples illustrate several characteristics of query formatting.

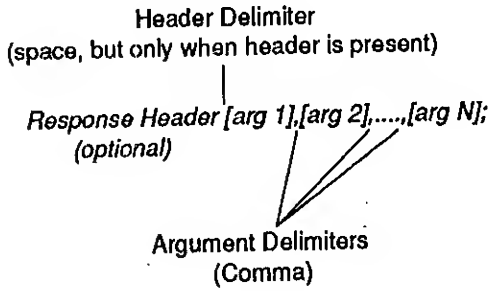
- A question mark (?) must follow the query header
- When an argument is used, it must be separated from the question mark by a space
- A command header can often be (but not always) turned into a query by adding a question mark (?)





## Responses

A response can be represented graphically as shown in this example.




With the exceptions of the SET?, PLOT?, and WAVFRM? queries (which never produce response headers), response headers are optional. Headers are turned on and off with the HDR command. When HDR is ON, all responses (except SET?, PLOT?, and WAVFRM?) have headers.

No responses have headers when HDR is OFF. Further, when HDR is OFF, the link in the responses to marker queries with linked numeric arguments (such as MAMpl? or MFReq?) is also turned off. Response headers cannot be selectively suppressed unless HDR OFF is set before the response and then HDR is set to ON again after the response.

Most responses consist of an optional header and the response argument. However, responses such as the WFMPRE? response have many arguments separated by commas. Others, including the response to CURVE?, contain hundreds of data words in a single argument called a binary block. SET? is a special query that returns many arguments separated by semicolons (;) so the response can be read back to the spectrum analyzer as a series of commands.

Note that a response always terminates with a semicolon (or line feed if MSGDLM is set to LF).

Following are examples of three queries and their resulting responses. The first line after each query is the response with the headers on; the second line is the response to these queries with the headers off.

```
FREQ?
    FREQ 200.00E+6;
    200.00E+6;
VIEW?
    VIEW WATERFALL:OFF,A:ON,B:OFF,
        C:OFF,D:ON,MINUSA:OFF;
    WATERFALL:OFF,A:ON,B:OFF,
        C:OFF,D:ON,MINUSA:OFF;
MAMPL? DELTA
    MAMPL DELTA:18.5;
    18.5;
```

## Headers

Examples of headers in this manual are written in capital letters, but the 2711 and 2712 understand lower-case letters (or a mixture of upper- and lower-case) as well. Furthermore, our examples use the long form of command and query headers. However, most (but not all) command and query headers can be written using as few as the first three letters. That is, `FRE` means the same as `FREQ` and `TTLM` means the same as `TTLMODE`.

As you become more familiar with the commands and queries, you will find that the shortened forms are quicker to use. However, the long forms are easier to read and are always used by the instrument in response headers.

Throughout the remainder of this manual the long form for headers will be used, but we will print the required letters in capitals and the optional characters of the longer form in lower-case letters. The following examples illustrate how headers will appear in this manual.

```
FREq
VRTdsp
MAMpl
CALSig
```

You can also use variations of the short and long header forms. That is, `VRT`, `VRTd`, `VRTds`, and `VRTdsp` are all acceptable header forms for the vertical display command or query.





## Space ( )

The space character is used to separate a command or response header from its first argument, or to separate the question mark following a query header from its argument. The space is optional when there are no arguments.

## Comma (,)

Commas are used to separate or delimit multiple arguments in commands and responses. They should not be used elsewhere.

## Semicolon (;)

Semicolons are normally used to separate or delimit multiple message units in a single message. They may also (optionally) follow the last argument of a command or query. They should not be used elsewhere. The line feed character can be optionally substituted for the semicolon.

## Colon (:)

Colons are used to connect the two parts of a linked argument. They should not be used elsewhere.

## Line Feed (GPIB)

The line feed character can be used instead of a semicolon to delimit message units in a single message. The 2711 and 2712 will substitute line feeds for semicolons in its responses when `MSGdlm` is set to LF. Line feed can also be used as a message terminator with controllers that do not support the GPIB EOI protocol. These two uses are separate and not exclusive.

## Carriage Return and Line Feed (RS-232)

When a controller sends data to the 2711 and 2712, the spectrum analyzer interprets CR, LF, or CR LF as a message terminator. Messages sent by the spectrum analyzer over the RS-232 interface can be terminated in the following ways.

- By CR only (carriage return, ASCII 13)
- By LF only (line feed, ASCII 10)
- By CR-LF (carriage return followed by line feed)

# Section 3 — Functional Groups

Replace this page with the tab divider of the same name.

# SECTION 3
# FUNCTIONAL GROUPS

The instrument-specific commands and queries have been listed several different ways in this manual to make them more convenient to the user. The level of detail increases as you progress through the manual.

Table 3-1 provides a quick reference that contains all instrument-specific commands and queries. Tables 3-2 to 3-16 relate each GPIB command to a corresponding front panel control or menu. Section 4, ***Command and Query Definitions***, provides a detailed description of each command and query. Section 4 includes all commands and queries available for the 2711 and 2712 listed in alphabetical order with a discussion of each, syntax examples, data formats, and other useful information.

The capital letters in a header indicate the minimum number of letters that must be supplied for the spectrum analyzer to recognize the header. For instance, the query `ACQ?` would produce the same response as the query `ACQmode?`. The lower-case letters indicate optional additional letters which may be used to clarify the meaning of the header. The spectrum analyzer accepts either upper- or lower-case letters, but it WILL NOT accept headers that contain letters other than those indicated in the following tables. For instance, if the tables show a command in this form,

```
CMEas
```

you can enter the command in any combination of upper- and lower-case letters as in these examples.

```
CME
CmEA
cMeas
```

However, attempts to enter the command using a different combination of letters, as in these examples, will be ignored. An SRQ and an error message will be generated. See Section 5, ***Status Reporting***, for additional information.

```
CMEASURE
cma
```





## THE COMMAND/QUERY LIST

Table 3-1 lists all the commands available for controlling the 2711 and 2712 Spectrum Analyzers. This table shows the experienced user the correct form of each command and query header. Table 3-1 is a convenient reference for the mnemonics of each command or query for users who are already familiar with instrument functions.

**Table 3-1. Commands and Queries.**

| | | | |
|---|---|---|---|
| ACQmode | CLOck? | DLValue? | INIT |
| ACQmode? | CLRKey | DSRc | KEY |
| AQP | CLRMenu | DSRc? | KEY? |
| AQP? | CMEas | EMC | LRAmpl |
| AREs | CNBw | EMC? | MAMpl? |
| AREs? | CNBw? | EOS | MARker |
| ARFatt | CNMode | EOS? | MARker? |
| ARFatt? | CNMode? | ERAse | MEMory? |
| ATBl? | CNResult? | ERr? | MEXchg |
| ATHrhld | CNTtrak | EVEnt? | MFReq |
| ATHrhld? | CNTtrak? | FILE | MFReq? |
| AVDest | COUnt? | FILE? | MHDest |
| AVDest? | CREs | FINe | MHDest? |
| AVG | CREs? | FINe? | MKTime |
| AVG? | CURve | FOFset | MKTime? |
| AVMode | CURve? | FOFset? | MLFtnxt |
| AVMode? | DATe | FOMode | MMAx |
| AVNum | DATe? | FOMode? | MNHld |
| AVNum? | DATIme? | FREq | MNHld? |
| BWMode | DEFMenu | FREq? | MPOs? |
| BWMode? | DETector | GRAt | MRGTnxt |
| BWNum | DETector? | GRAt? | MSGdlm |
| BWNum? | DIR? | GTL | MSGdlm? |
| BWResult? | DIScor | HDR | MSTep |
| CALSig | DIScor? | HDR? | MTUNE |
| CALSig? | DLIne | HELp? | MVPos? |
| CENsig | DLIne? | HRAmpl | MXHld |
| CFSF | DLLimit | ID? | MXHld? |
| CFSF? | DLLimit? | IMPCor | MXRlvl |
| CLOck | DLValue | IMPCor? | MXRlvl? |

| | | | |
|---|---|---|---|
| MXSpn | RFAtt | TABle | TVLMode |
| MXSpn? | RFAtt? | TABle? | TVLMode? |
| NNBw | RLUnit | TAMpl? | TVLStd |
| NNBw? | RLUnit? | TEXt | TVLStd? |
| NNMode | ROFset | TEXt? | VDMode |
| NNMode? | ROFset? | TFReq? | VDMode? |
| NNResult? | ROMode | TGEnab | VFEnab |
| NORM | ROMode? | TGEnab? | VFEnab? |
| NORM? | RQS | TGLevel | VFMode |
| OBWMode | RQS? | TGLevel? | VFMode? |
| OBWMode? | RS232 | TGMan | VIDflt |
| OBWPcnt | RS232? | TGMan? | VIDflt? |
| OBWPcnt? | RTIme | TGOMode | VIEw |
| OBWResult? | RTIme? | TGOMode? | VIEw? |
| PKHeight | SAVe | TGOOffset | VMAnttbl |
| PKHeight? | SAVe? | TGOOffset? | VMAnttbl? |
| PLLmode | SET? | TGTMode | VMDEst |
| PLLmode? | SGErr | TGTMode? | VMDEst? |
| PLOT? | SGErr? | TGTRack | VMDIst |
| POFset | SGSrch | TGTRack? | VMDIst? |
| POFset? | SGTrak | THRhld | VMMkrunit |
| PRDouts? | SGTrak? | THRhld? | VMMkrunit? |
| PREamp | SIGswp | TIMe | VMOnitor |
| PREamp? | SIGswp? | TIMe? | VMOnitor? |
| PROTset | SPAn | TIMMode | VPOlarity |
| PROTset? | SPAn? | TIMMode? | VPOlarity? |
| PSTep | SSBegin | TITLe | VRTdsp |
| PTYpe | SSBegin? | TITLe? | VRTdsp? |
| PTYpe? | SSEnd | TMOde | VSYnc |
| QPFilt | SSEnd? | TMOde? | VSYnc? |
| QPFilt? | SSResult? | TOPsig | WAIt |
| RECall | STByte? | TRIgger | WAVfrm? |
| REDout | STEp | TRIgger? | WFMpre |
| REDout? | STEp? | TTLMode | WFMpre? |
| REFlvl | STOre | TTLMode? | ZERosp |
| REFlvl? | STPinc | TUNe | ZERosp? |
| RESbw | STPinc? | TVLine | |
| RESbw? | STStop | TVLine? | |





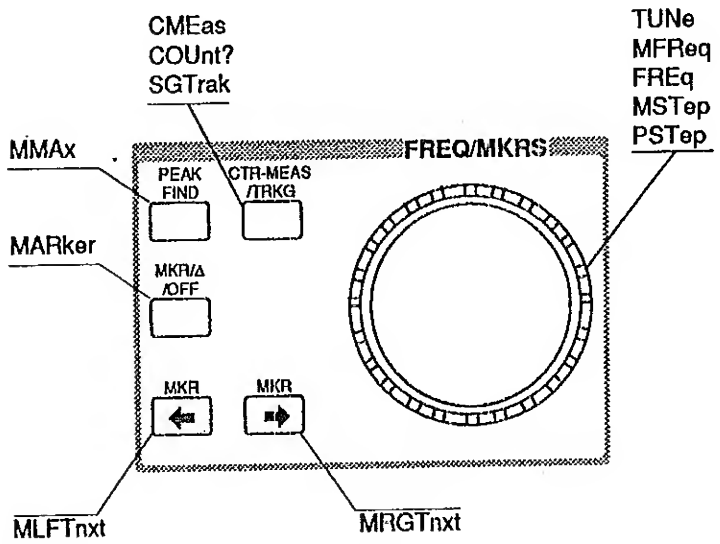
## COMMAND AND QUERY FUNCTIONAL GROUPS

Tables 3-2 through 3-16 show how the commands and queries available for programming the spectrum analyzer correspond to the front panel controls and menu selections. An illustration of each front panel function block or menu is shown. Related commands are placed beside the feature that it controls.

The functional groupings and menus appear in the following order.

- FREQ/MKRS function block
- MKR/FREQ Menu
- FREQUENCY-SPAN/DIV-REF LEVEL function block
- VERT SCALE function block and PLOT and READOUT controls
- INPUT Menu
- SWP/TRIG Menu
- SWEEP and RES BW function blocks
- DISPLAY STORAGE function block
- DISPL Menu
- APPL Menu
- UTIL Menu
- DEMOD/TG Menu
- Curve and Waveform commands
- System-related commands
- Miscellaneous commands

**Table 3-2. FREQ/MKRS Front Panel Commands.**

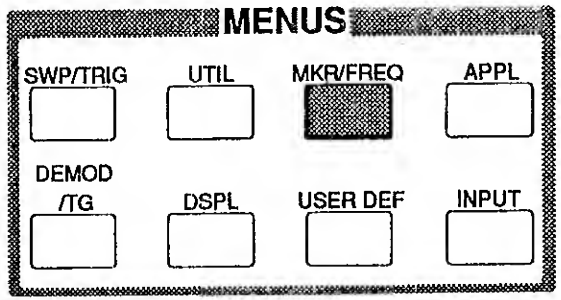



| Header | Function |
|---|---|
| CMEas | Perform a center measure. |
| COUnt? | What is the counter reading? (2711 requires Option 02 Frequency Counter) |
| FREq | Set the start or center frequency. |
| FREq? | What is the start or center frequency? |
| MARker | Turn one or both markers on and off. |
| MARker? | What is the current marker status? |
| MFReq | Set the marker frequency. |
| MFReq? | What is the frequency of either or both markers? |
| MLFtnxt | Move the marker to the next signal peak left. |
| MMAx | Move the marker to highest data point on screen. |
| MRGTnxt | Move the marker to the next signal peak right. |
| MSTep | Equivalent to turning the knob 1 click to the left. |
| PSTep | Equivalent to turning the knob 1 click to the right. |
| SGTrak | Turn signal tracking on and off. |
| SGTrak? | Is signal tracking on or off? |
| TUNe | Change frequency. |





**Table 3-3. MKR/FREQ Menu commands.**




| MARKER/FREQUENCY MENU | | |
|---|---|---|
| 0 THRESHOLD | AUTO -16.3DBM | THRhld, ATHrhld |
| 1 PROGRMD TUNING INC | | STEp |
| 2 KNOB FUNCTION | MARKER | TMOde |
| 3 MARKER TO REFERENCE LEVEL | | TOPsig |
| 4 MOVE MARKER TO NEXT PEAK | | LRAmpl, HRAmpl |
| 5 TRANSPOSE MARKERS | | MEXchg |
| 6 MARKER START/STOP | | STStop |
| 7 FREQUENCY START/STOP | | |
| 8 TUNING INCREMENT | AUTO | STPinc |
| 9 SETUP TABLE | | |
| 0 CENTER/START FREQ | CENTER | CFSF |
| 1 COUNTER RESOLUTION | 1HZ | CREs, CNTrak |
| 2 TABULAR TUNING TABLES | | TABle |
| 3 FREQ OFFSET | 0.000HZ | FOFfset |
| 4 FREQ OFFSET MODE | OFF | FOMode |

`MAMpl?`, `MKTime?`, `TAMpl?`, and `TFReq?` return on-screen measurement parameters. `MPOs?` and `MVPos?` have no visible affect when the spectrum analyzer has an analog display (all Display Storage registers disabled).

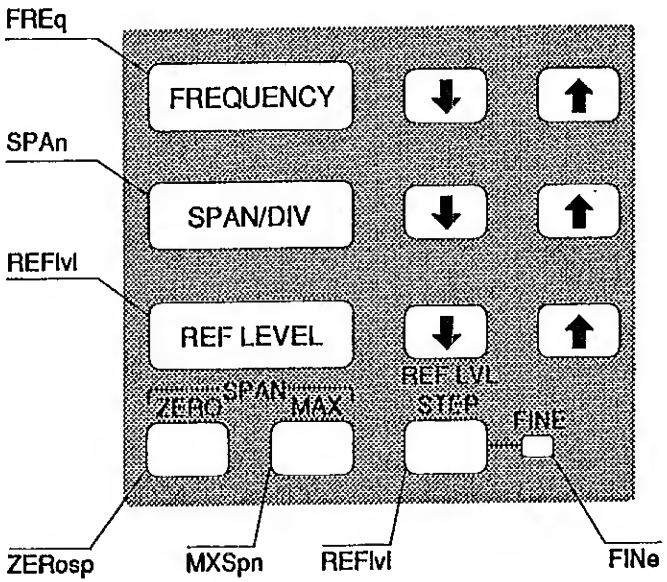
| Header | Function |
|---|---|
| ATHrhld | Turn the auto threshold on and off. |
| ATHrhld? | Is the auto threshold on or off? |
| CENsig | Set frequency to the marker frequency. |
| CFSF | Select center or start frequency. |
| CFSF? | Is center or start frequency being used? |
| CNTtrak | Turn counter on and off during signal track. [1] |
| CNTtrak? | Is counter on or off during signal track? [1] |
| CREs | Set the counter resolution. [1] |
| CREs? | What is the counter resolution? [1] |
| FOFfset | Set the frequency offset. |
| FOFfset? | What is the frequency offset? |
| FOMode | Turn frequency offset mode on and off. |
| FOMode? | Is frequency offset mode on or off? |
| HRAmpl | Move the marker to next higher amplitude peak. |
| LRAmpl | Move the marker to the next lower amplitude peak. |
| MAMpl? | What is the amplitude of either or both markers? |
| MEXchg | Exchange markers. |
| MKTime | Set the marker time in zero span mode. |
| MKTime? | What is the time of either or both markers? |
| MTUNE | Change marker frequency by a specified amount. |
| MPOs? | What is the hor. position of either or both markers? |
| MVPos? | What is the vert. position of either or both markers? |
| STEp | Set the frequency increment step size. |
| STEp? | What is the frequency increment step size? |
| STPinc | Set the type of frequency increment. |
| STPinc? | What type of frequency increment is being used? |
| STStop | Set start / stop frequencies to marker frequencies. |
| TABle | Select tabular tuning table. |
| TABle? | What tabular tuning table is selected? |
| TAMpl? | What is amplitude of tracked signal? |
| TFReq? | What is frequency of tracked signal? |
| THRhld | Replace the auto threshold with the specified value. |
| THRhld? | What is the threshold value? |
| TMOde | Select the knob function. |
| TMOde? | What is the knob function. |
| TOPsig | Change reference level to the marker amplitude. |

[1] 2711 requires Option 02 (Frequency Counter).



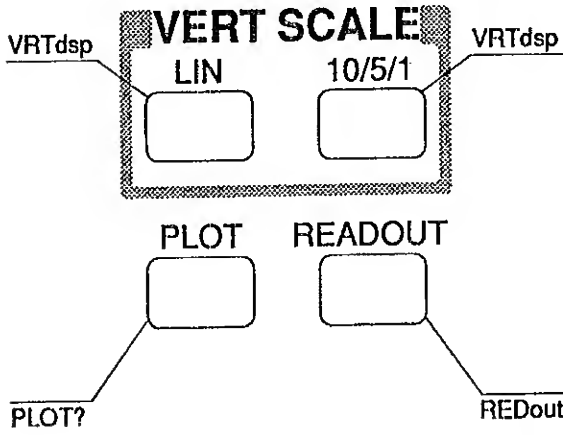


**Table 3-4. FREQUENCY, SPAN/DIV, and REF LEVEL Front Panel Commands.**




| Header | Function |
|---|---|
| FINe | Selects 1 dB or 10 dB reference level steps. |
| FINe? | Is the reference level step 1 dB or 10 dB?. |
| FREQ | Set the center or start frequency. |
| FREQ? | What is the current center or start frequency? |
| MXSpn | Turn MAX SPAN mode on and off. |
| MXSpn? | Is MAX SPAN on or off? |
| REFlvl | Set/increment/decrement reference level. |
| REFlvl? | What is the reference level? |
| SPAn | Select the frequency span per division. |
| SPAn? | What is the frequency span? |
| ZERosp | Turn ZERO SPAN on and off. |
| ZERosp? | Is ZERO SPAN on or off? |

Table 3-5. VERT SCALE, PLOT, READOUT Front Panel Commands.

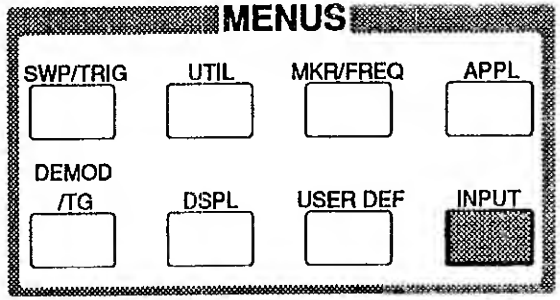



| Header | Function |
|---|---|
| PLOT? | Return screen plot data from the spectrum analyzer. |
| REDout? | Turn the on-screen readouts on or off. |
| REDout? | Are on-screen readouts on or off? |
| VRTdsp | Select the vertical scale factor. |
| VRTdsp? | What is the vertical scale factor? |





Table 3-6. INPUT Menu Commands.




INPUT MENU

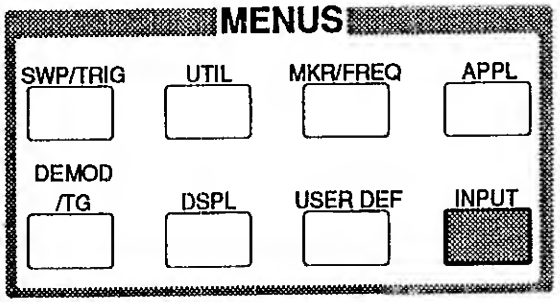
| | | |
|---|---|---|
| 1 PREAMP | OFF | PREamp |
| 2 50 OHM DBM/75 OHM DBMV | 50 | IMPCor |
| 3 REF LEVEL UNIT | DBUVM | RLUnit |
| 4 1ST MXR INPUT LVL | -30DBM | MXRlvl |
| 5 RF ATTENUATION | AUTO 50DB | ARFatt, RFAtt |
| 6 EXTERNAL ATTEN/AMPL | NONE | ROFset, ROMode |
| 9 CAL SIG @ 100MHZ -30DBM | OFF | CALsig |

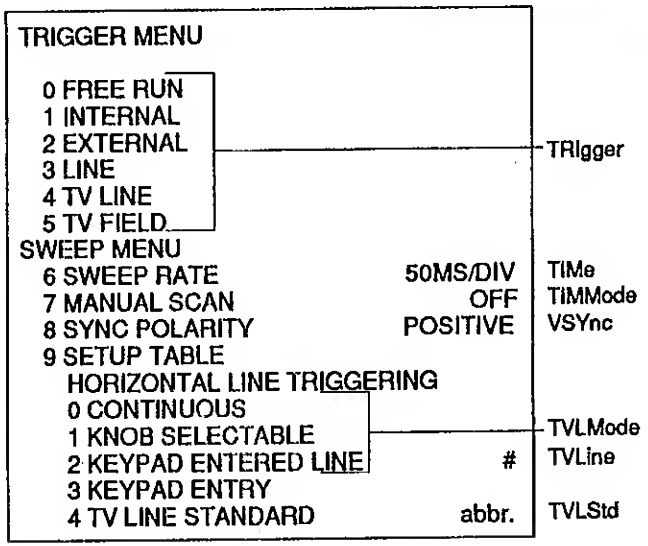
| Header | Function |
|---|---|
| ARFatt | Turn auto RF attenuation on and off. |
| ARFatt? | Is auto RF attenuation on or off? |
| ATBl? | Provide a listing of an antenna correction table. |
| CALsig | Turn the internal calibration signal on and off. |
| CALsig? | Is the internal calibration signal on or off? |
| IMPCor | Corrects indicated amplitude for 50/75 ohm source. |
| IMPCor? | Amplitude corrected for 50 or 75 ohm source? |
| MXRlvl | Select first mixer level. |
| MXRlvl? | What is first mixer level? |
| PREamp | Turn the preamp on and off. |
| PREamp? | Is the preamp on or off? |
| RFAtt | Set the RF attenuation to a specific value. |
| RFAtt? | What is the RF attenuation? |
| RLUnit | Select reference level unit. |
| RLUnit? | What is the reference level unit? |
| ROFset | Set the reference level offset and turn it on and off. |
| ROFset? | What is the reference level offset? |
| ROMode | Turn reference level offset mode on and off. |
| ROMode? | Is reference level offset mode on or off? |
| VMAnttbl | Select an antenna table. |
| VMAnttbl? | What antenna table is selected? |
| VMDIst | Select measurement distance in dBµV/m mode. |
| VMDIst? | What is the measurement distance? |
| VMDEst | Select destination register in dBµV/m mode. |
| VMDEst? | What is the destination register? |
| VMMkrunit | Select marker units of dBµV/m or Volts/m in dBµV/m mode. |
| VMMkrunit? | What is the marker unit in dBµV/m mode? |





**Table 3-7. SWP/TRIG Menu Commands.**



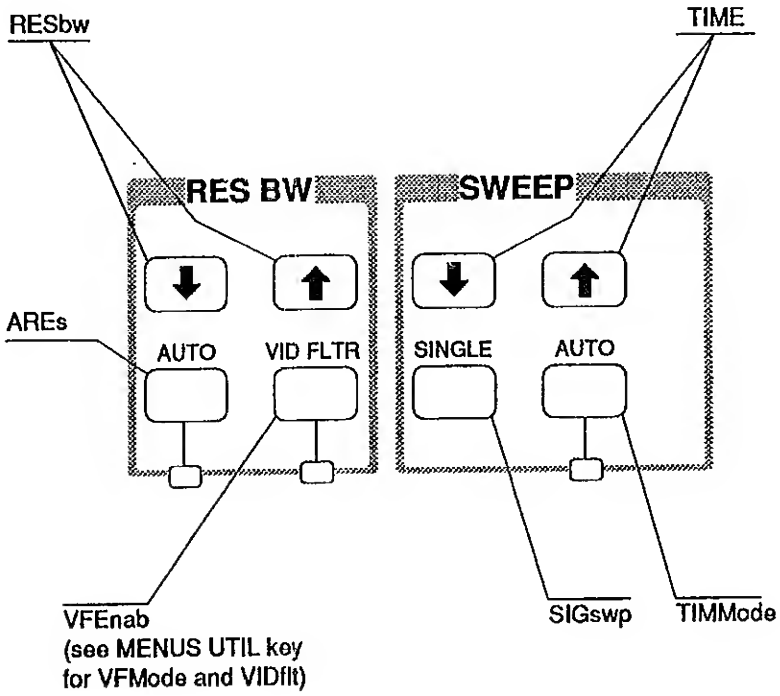
| Header | Function |
|---|---|
| TIMe | Set the sweep rate. |
| TIMe? | What is the sweep rate? |
| TIMMode | Select auto, manual, or programmed sweep. |
| TIMMode? | What is the sweep mode? |
| TRIgger | Select the trigger mode. |
| TRIgger? | What is the trigger mode? |
| TVLIne | Select the number of the video raster line to trigger on when TV line triggering is selected. [1] |
| TVLine? | What is the number of the TV line to trigger on? [1] |
| TVLMode | Selects continuous or programmed TV line trigger. |
| TVLMode? | Is continuous or programmed TV line trigger used? |
| TVLStd | Selects TV standards used in various countries. |
| TVLStd? | What TV standard is being used? |
| VSYnc | Selects positive or negative video sync polarity. |
| VSYnc? | Is positive or negative sync polarity being used? |

[1] Requires Video Monitor mode (Option 10).



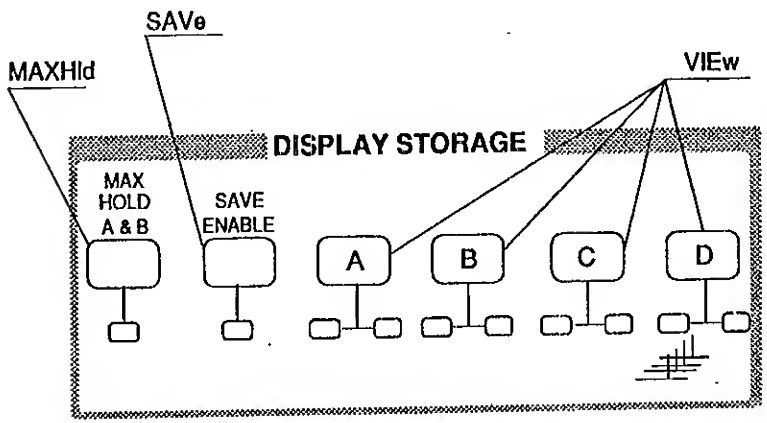


**Table 3-8. SWEEP and RES BW Front Panel Commands.**




| Header | Function |
|---|---|
| AREs | Turn AUTO resolution bandwidth on and off. |
| AREs? | Is AUTO resolution bandwidth on or off? |
| SIGswp | Selects and arms the single sweep mode. |
| SIGswp? | What is the status of the single sweep mode? |
| TIMe | Select/increment/decrement the sweep speed. |
| TIMe? | What is the sweep speed? |
| TIMMode | Select auto, manual, or programmed sweep mode. |
| TIMMode? | What sweep mode is selected? |

**Table 3-9. DISPLAY STORAGE Front Panel Commands.**

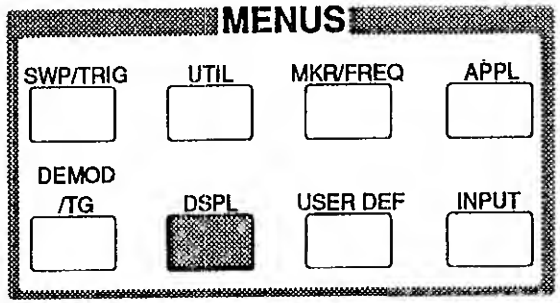



| Header | Function |
|---|---|
| MXHld | Turn max hold function on and off. |
| MXHld? | Is the max hold function on or off? |
| SAVe | Turn display storage on or off in any or all registers. |
| SAVe? | Is storage on or off in any or all registers? |
| VIEw | Turn display on and off in any or all registers. Also turns waterfall and B,C minus A modes on and off. |
| VIEw? | What is the display status of any or all registers? |





**Table 3-10. DSPL Menu Commands.**



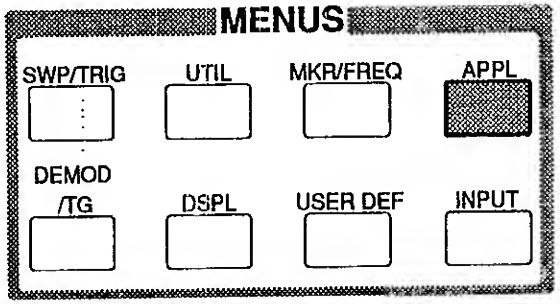

| DISPLAY MENU | | |
|---|---|---|
| 0 DIGITAL/ANALOG | DIGITAL | |
| 1 ENSEMBLE AVERAGING | | |
| 1 INITIATE AVERAGING | | AVG |
| 2 TERMINATE AVERAGING | | AVG |
| 3 MAX | | AVMode |
| 4 MEAN | | |
| 5 MIN | | |
| 6 MAX/MIN | | |
| 7 NUMBER OF AVERAGES | # | AVNum |
| 8 SAVE RESULTS IN DISPLAY | C | AVDest |
| 2 B,C MINUS A | OFF | VIEw Minusa: |
| 3 B,C OFFSET TO | CENTER | POFset |
| 4 ACQUISITION MODE | PEAK | ACQmode |
| 5 TITLE MODE | OFF | TITLe, TTLMode |
| 6 GRATICULE ILLUMINATION | ON | GRAt |
| 7 DISPLAY SOURCE (AM) | INTERNAL | DSRc |
| 8 DISPLAY LINE | | |
| 1 ON/OFF | OFF | DLIne |
| 2 VALUE ENTRY | 20.0DBM | DLValue |
| 3 DISPLAY LINE TO MARKER | | DLValue |
| 4 LIMIT DETECTOR | OFF | DLLimit |
| 9 MIN HOLD IN WFM C | OFF | MNHld, MHDest |

| Header | Function |
|---|---|
| ACQmode | Selects peak or max/min acquisition mode. |
| ACQmode? | What is the acquisition mode? |
| AVDest | Select destination register for ensemble averaging. |
| AVDest? | What is the destination register for averaging? |
| AVG | Turn ensemble averaging on and off. |
| AVG? | Is ensemble averaging on or off? |
| AVMode | Select the ensemble averaging mode. |
| AVMode? | What is the ensemble averaging mode? |
| AVNum | Select number of sweeps averaged . |
| AVNum? | What is the number of sweeps averaged? |
| DLIne | Turn the display line on and off. |
| DLIne? | Is the display line on or off? |
| DLLimit | Control the limit detector. |
| DLLimit? | What is the limit detector status? |
| DLValue | Set the display line value and turn it on. |
| DLValue? | What is the display line value? |
| DSRc | Select the detection mode. |
| DSRc? | What is the detection mode? |
| GRAt | Turn the graticule light on and off. |
| GRAt? | Is the graticule light on or off? |
| MNHld | Turn min hold function on and off |
| MNHld? | Is the min hold function on or off? |
| MHDest | Select the min hold destination waveform. |
| MHDest | What is the min hold destination waveform? |
| POFset | Offset B,C-A mode to center or top of screen. |
| POFset? | Is B, C -A offset to top or center of screen? |
| TEXt | Display the indicated text on line 8 of display. |
| TEXt? | What is the text string being displayed? |
| TITLe | Display the indicated text as a title in title mode. |
| TITLe? | What is the title? |
| TTLMode | Turn title mode on and off. |
| TTLMode? | Is title mode on or off? |
| VIEw Minusa: | Turn B,C MINUS A mode on and off. |





**Table 3-11. APPL Menu Commands.**




| APPLICATION MENU | | |
|---|---|---|
| 0 BANDWIDTH MODE @ | -3DBC | BWMode |
| 1 CARRIER TO NOISE @ | 5.0MHZBW | CNMode |
| 2 NOISE NORM'D @ | 1.0HZBW | NNMode |
| 3 SIGNAL SEARCH MENU | | |
| 0 BEGIN FREQ | 88.000MHZ | SSBegin |
| 1 END FREQ | 108.000MHZ | SSEnd |
| 2 START TEST | | SGSrch |
| 3 DISPLAY RESULTS | # SIGNALS | SSResult |
| 4 OCCUPIED BW @ | 99% | OBWpcnt |
| 5 EMC MODE[1] | | EMC |
| 6 QUASI-PEAK[1] | 200HZ FLTR | AQP, DSRc |
| 7 FM DEVIATION MODE | | |
| 9 SETUP TABLE | | |
| 0 DB DOWN FOR BW MODE | -3DBC | BWNum |
| 1 NORM BW FOR C/N | 5.0MHZBW | CNBw |
| 2 NOISE NORM'D BW | 1.0HZBW | NNBw |

`BWResult?`, `CNResult?`, and `NNResult?` return results normally displayed on screen.

**[1] Only available with 2712 Option 12 (Quasi-Peak Detector).**

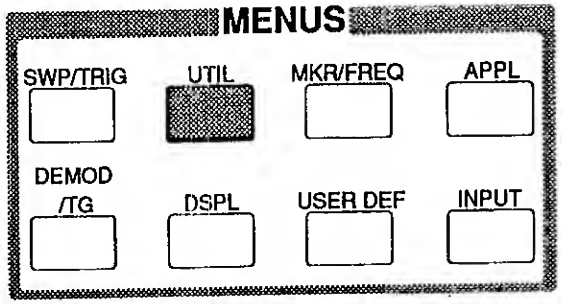
| Header | Function |
|---|---|
| AQP | Turn auto quasi-peak mode on and off.[1] |
| QPFilt | Select quasi-peak detector band for manual mode.[1] |
| BWMode | Turn bandwidth mode on and off. |
| BWMode? | Is bandwidth (BW) mode on or off? |
| BWNum | Set the number of dB down for BW mode. |
| BWNum? | What is the dB down setting in BW mode? |
| BWResult? | What is the BW at the specified dB down? |
| CNBw | Set noise BW for carrier-to-noise (C/N) mode. |
| CNBw? | What is the noise BW in C/N mode? |
| CNMode | Turn carrier-to-noise mode on and off. |
| CNMode? | Is carrier-to-noise mode on or off? |
| CNResult? | What is the C/N ratio? |
| DSRc | Set the detection mode. |
| DSRc? | What is the detection mode? |
| EMC | Set EMC mode on or off.[1] |
| EMC? | Is EMC mode on or off.[1] |
| NNBw | Set the noise BW for normalized noise mode. |
| NNBw? | What is the noise BW in normalized noise mode? |
| NNMode | Turn normalized noise mode on and off. |
| NNMode? | s normalized noise mode on or off? |
| NNResult? | What is the normalized noise in the specified BW? |
| OBWMode | Set occupied bandwidth mode to on, off or idle. |
| OBWMode? | Is the occupied bandwidth mode on, off or idle? |
| OBWPcnt | Set percent (1 to 99%) occupied bandwidth. |
| OBWPcnt? | Return the current occupied bandwidth percent. |
| OBWResult? | Return the results (Hz) of the most recent occupied bandwidth measurement. |
| SSBegin | Set the beginning signal search frequency. |
| SSBegin? | What is the beginning signal search frequency? |
| SSEnd | Set the ending signal search frequency. |
| SSEnd? | What is the ending signal search frequency? |
| SGSrch | Search for signals greater than threshold (THRhld) between beginning and ending search frequencies. |
| SSResult? | What is the result of the signal search? |

[1] Only available with 2712 Option 12 (Quasi-Peak Detector).





Table 3-12. UTIL Menu Commands.




| UTILITY MENU | | |
|---|---|---|
| 0 INITIALIZE INSTR SETTINGS | INIT | RECall |
| 1 STORED SETTINGS/DISPLAYS | | STOre |
| 2 KEYPAD ENTERED SETTINGS | | ERAse |
| 5 VIDEO FILTER | WIDE | VFMode, VIDfit |
| 3 NORMALIZATIONS | | NORM |
| 4 SYSTEM CONFIGURATION | | |
| 1 SCREEN PLOT CONFIGURATION | | PTYpe |
| 3 INSTRUMENT CONFIGURATION | | |
| 1 MINIMUM SIGNAL SIZE | 20 | PKHeight |
| 4 PHASELOCK[1] | ON | PLLmode |
| 5 FREQUENCY CORRECTIONS | ON | ACQmode |
| 4 REAL-TIME CLOCK SETUP | | DIScor |
| 0 SET DAY | | |
| 1 SET MONTH | | |
| 2 SET YEAR | | DATe |
| 3 SET HOUR | | RTIme |
| 4 SET MINUTE | | |
| 5 SET SECONDS TO :00 | | |
| 6 DISPLAY DATE/TIME | ON | CLOck |
| 5 STORED SETTINGS PROTECT | OFF | PROTset |
| 6 FILE SYSTEM DIRECTORY | | DIR? |
| 9 INSTALLED OPTIONS DISPLAY | | ID? |
| 5 INSTR DIAGNOSTICS/ADJUSTMENTS | | |
| 7 SERVICE REQUEST | | RQS |

[1] Only available with 2712.

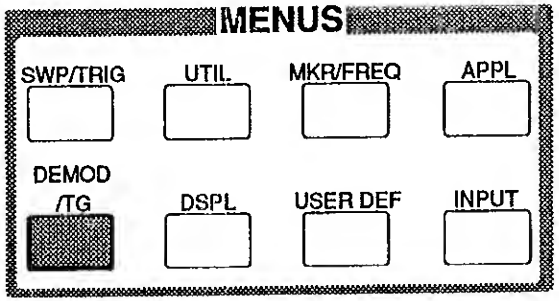
| Header | Function |
|---|---|
| DIR? | Return a spectrum analyzer file system directory. |
| DIScor | Turn the frequency corrections on and off. |
| DIScor? | Are the frequency corrections on or off? |
| DATe | Set the real-time clock date.[2] |
| DATe? | What is the real-time clock date? |
| DATIme? | What is the time of day? |
| CLOck | Turn the date and time display on or off. |
| CLOck? | Is the date and time display on or off? |
| ERAse | Erase the stored settings in a particular register. |
| ID? | List the spectrum analyzer firmware version and installed options. |
| INIT | Reset to user-defined or factory power-up settings. |
| NORM | Carry out the indicated normalizations. |
| NORM? | Return a list of current normalization parameters. |
| PKHeight | Set the minimum signal height for marker functions. |
| PKHeight? | What is min signal height for marker functions? |
| PLLmode | Turn phase lock on and off.[1] |
| PLLmode? | Is phase lock on or off?[1] |
| PROTset | Turn stored settings files protection on and off. |
| PROTset? | Is stored settings files protection on or off? |
| PTYpe | Specify the plotter type for screen plots. |
| PTYpe? | What is the specified plotter type? |
| RECall | Recall a stored settings file. |
| RTIme | Set the real-time clock time. |
| RTIme? | What is the real-time clock time? |
| STOre | Store the current settings in a stored settings file. |
| VFMode | Selects auto or fixed video filter mode. |
| VFMode? | Is auto or fixed video filter mode selected? |
| VIDfIt | Sets and turns the video filter on and off. |
| VIDfIt? | What video filter is selected? |

[1] Only available with 2712.





**Table 3-13. DEMOD/TG Menu Commands.**




| DMOD/TG MENU | | |
|---|---|---|
| 0 OFF | | |
| 1 AM DEMODULATOR | | DETector |
| 2 FM DEMODULATOR | | |
| 3 BROADCAST (AM) | VIDEO | VMOnitor |
| 4 TRACKING GENERATOR[1] | OFF | TGEnab |
| 5 TG FIXED LEVEL[1] | # | TGLevel |
| 6 TG VARIABLE LEVEL[1] | OFF | TGMan |
| 7 TG TRACKING[1] | OFF | TGTMode, TGTRack |
| 8 TG EXT ATTEN/AMPL[1] | # | TGOMode, TGOOffset |
| 9 VIDEO SETUP MENU | | |
| 0 VIDEO DETECT MODE[2] | BROADCAST | VDMode |
| 1 SYNC POLARITY | POSITIVE | VSYnc |
| 2 VIDEO POLARITY | NEGATIVE | VPOlarity |

[1] Only available with Option 04 (Tracking Generator).

[2] Only available with Option 10 (Video Monitor).

| Header | Function |
|---|---|
| DETector | Turn on/select which audio detector is used. |
| DETector? | Which audio detector is being used? |
| TGEnab | Turns the tracking generator on and off.[1] |
| TGEnab? | Is the tracking generator on or off?[1] |
| TGLevel | Sets the tracking generator output level.[1] |
| TGLevel? | What is the tracking generator output level?[1] |
| TGMan | Enables and disables manual control of tracking gen.[1] |
| TGMan? | Is manual tracking gen. control enabled or disabled?[1] |
| TGOMode | Turn tracking generator output level offset on or off.[1] |
| TGOMode? | Is tracking generator output level offset on or off?[1] |
| TGOOffset | Sets the tracking generator output level offset.[1] |
| TGOOffset? | What is the tracking generator output level offset?[1] |
| TGTMode | Turns the tracking generator tracking on and off.[1] |
| TGTMode? | Is the tracking generator tracking on or off?[1] |
| TGTRack | Sets the tracking generator tracking.[1] |
| TGTRack? | What is the tracking generator tracking?[1] |
| VDMode | Selects broadcast or satellite video demodulation.[2] |
| VDMode? | Is broadcast or satellite video demodulation used?[2] |
| VMOnitor | Turns the video monitor on or off. |
| VMOnitor | Is the the video monitor on or off? |
| VPOlarity | Selects positive or negative video polarity. |
| VPOlarity? | Is positive or negative video polarity selected? |
| VSYnc. | Selects positive or negative video sync polarity. |
| VSYnc? | Is positive or negative sync polarity being used? |

[1] Only available with Option 04 (Tracking Generator).

[2] Only available with Option 10 (Video Monitor).





**Table 3-14. Curve and Waveform Commands.**

| Header | Function |
|---|---|
| CURve | Transfer waveform data to the register specified by WFMpre, using the encoding set by WFMpre. |
| CURve? | Transfer waveform data from the specified register or from the register set with WFMpre, using the encoding specified by the WFMpre command. |
| WAVfrm? | Same as WFMPRE? followed by CURVE? |
| WFMpre | Specifies source or destination register used for transferring waveform data with the CURve command or query. Also, specifies the encoding to be used on the waveform data. |
| WFMpre? | Request the complete waveform preamble or ask which register and encoding are to be used for waveform transfers. |

Waveform transfers are not reflected on any menu or function block. They transfer data representing on-screen spectra and their formatting between the spectrum analyzer and the controlling computer.

Table 3-15. System-related Commands.

| Header | Function |
|---|---|
| EOS | Enable and disable SRQ on end-of-sweep. |
| EOS? | Is SRQ enabled or disabled at end-of-sweep? |
| ERr? | What is the error code? |
| EVEnt? | What is the event code? |
| GTL | Go to local operation. |
| HDR | Turns the response header on and off. |
| HDR? | Is the response header on or off? |
| HELp? | Spectrum analyzer sends a list of valid GPIB command headers. |
| MSGdlm | Selects semicolon or line feed as response delimiter. |
| MSGdlm? | What is the response delimiter? |
| RQS | Enables or disables SRQs (except power-on SRQ). |
| RQS? | Are SRQ's enabled or disabled? |
| RS232 | Set RS-232 communications parameters. |
| RS232? | Return all RS-232 communications parameters. |
| SET? | What are the current spectrum analyzer settings? |
| SGErr | Enable / disable SRQ when marker cannot find a signal. |
| SGErr? | Is marker-cannot-find-signal SRQ enabled or disabled? |
| STByte? | Return GPIB serial poll status byte. |
| WAIt | Command the spectrum analyzer to wait for the end of sweep. |

These commands and queries are independent of any spectrum analyzer menu or function block. They represent functions that effect the interaction of the spectrum analyzer and the GPIB controller or the RS-232 interface.





Table 3-16. Miscellaneous Commands.

| Header | Function |
|---|---|
| CLRMenu | Clear the menu on the spectrum analyzer screen. |
| CLRKey | Clear the last key pressed. |
| DEFMenu | Write a menu on the spectrum analyzer screen. |
| FILE | Store a binary block under a given file name. |
| FILE? | Return the named file as a binary block. |
| KEY | Simulate pressing a key. |
| KEY? | Return the identity of the last key pressed. |
| MEMory? | How much NVRAM is free? |
| PRDouts? | Return the spectrum analyzer on-screen readouts. |

The remaining commands include a set of miscellaneous commands and those which support on-screen menu definition and item selection. Refer to Section 6, ***Programming***, for an example of how these commands are used to create an interactive menu on the spectrum analyzer's display screen.

# Section 4 — Command/Query

Replace this page with the tab divider of the same name.

# SECTION 4
# COMMAND AND QUERY DEFINITIONS

This section contains an alphabetical listing of all instrument-specific commands and queries. The list defines each command or query. In addition, it contains all the information needed to send messages to the spectrum analyzer, or to interpret the responses from the spectrum analyzer.

## TYPOGRAPHICAL CONVENTIONS

Each spectrum analyzer command is discussed in the following format:

**COMmand** <arg> (if no argument is needed, <arg> is omitted)

Arguments: `Argument 1, argument 2, ...`
(If no argument is required, "None" is listed.)

Upper-case letters are required when entering data. Lower-case letters may be supplied if desired. Letters other than those shown will not be accepted by the spectrum analyzer.

Following each command is a general discussion of its arguments, specific precautions, and other important information.

Actual messages are shown in their correct syntax. When the number of possible messages is limited (such as commands that turn features on and off), all messages are shown as in the following example.

```
COMmand ON
COMmand OFF
```

Where there is a large range of arguments (such as numeric values), typical examples are shown as in this example.

```
COMmand 10.5 kHz (for example)
```

Typical examples are always followed by the phrase *(for example)*.





Each query is discussed in the following format.

**QUEry?** <arg> (In most cases no argument is needed, and <arg> is omitted.)

Arguments: `Argument 1, argument 2, ...`
(If no argument is required, "None" is listed.)

Following each query is a general discussion of its arguments, specific precautions, and other important information. A detailed description of the response to the query is also provided.

The query is shown along with its arguments. The spectrum analyzer response is shown indented on the following line. The response is always shown assuming that `HDR ON` is selected as in the following example.

```
QUEry?
    QUERY ON
    QUERY OFF
```

All responses are shown when the number of possible responses is limited (such as queries that report the on/off status of features). When a large range of responses is possible (such as numeric values), typical examples are shown as in this example.

```
QUERY?
    QUERY 10.500E+3 (for example)
```

Typical examples are always followed by the term *(for example)*.

# LIST OF COMMANDS AND QUERIES

The following list of commands and queries provides detailed information about the 2711 and 2712 instruction set. It does not attempt to explain the operation of the spectrum analyzer. Refer to the ***2711 Spectrum Analyzer User*** manual or ***2712 Spectrum Analyzer User*** manual for descriptions of the 2711 and 2712, or their features and functions.

**ACQmode** <arg>

Arguments: MAXMin, PEAK

This single-argument command designates the display storage acquisition mode.

```
ACQmode PEAK
ACQmode MAXMin
```

**ACQmode?**

Arguments: None

This simple query returns the currently selected acquisition mode.

```
ACQmode?
    ACQMODE PEAK
    ACQMODE MAXMIN
```

**AQP** <arg> (Only available for 2712 Option 12, Quasi-Peak Detector)

Arguments: ON, OFF

This single-argument command turns automatic Quasi-Peak mode on and off.

```
AQP ON
AQP OFF
```

**AQP?** (Only available for 2712 Option 12, Quasi-Peak Detector)

Arguments: None

This simple query returns the status of automatic Quasi-Peak mode.

```
AQP?
    AQP ON
    AQP OFF
```





**AREs** <arg>

Arguments: ON, OFF

This single-argument command turns automatic selection of resolution bandwidth on and off.

```
AREs ON
AREs OFF
```

**AREs?**

Arguments: None

This simple query returns the status of automatic resolution bandwidth selection mode.

```
AREs?
    ARES ON
    ARES OFF
```

**ARFatt** <arg>

Arguments: ON, OFF

This single-argument command turns automatic selection of RF attenuation on and off. The attenuation, linear scale factor, and reference level may change when auto selection is turned on, but not when auto selection is turned off.

```
ARFatt ON
ARFatt OFF
```

**ARFatt?**

Arguments: None

This simple query returns the status of automatic RF attenuation selection.

```
ARFatt?
    ARFATT ON
    ARFATT OFF
```

**ATBl?** <arg>

Arguments: None, integer in range of 1 to 5

This is a query with one or no argument that returns a listing of the specified antenna table. The argument is the number of the

antenna table to be listed. If a number outside the range is indicated, the last table in the range is returned (for instance, an argument of 6 returns table number 5, an argument of 0 returns table number 1). If no argument is specified, the currently selected table is returned.

```
ATBl? 3
    ATBL "ANTENNA 3
    Cal Distance = 3.0 Meters
    Frequency          Factor(dB)
    ------------------------------
    100.0MHz              1.0
    200.0MHz              2.0
    300.0MHz              3.0
       :                   :
    1.8GHz               18.0
    ------------------------------
    ";   (for example)
```

**ATHrhld**<arg>

Arguments: ON, OFF

This single-argument command turns automatic selection of signal threshold on and off. The threshold may change when turning on auto selection, but not when turning it off.

```
ATHrhld ON
ATHrhld OFF
```

**ATHrhld?**

Arguments: None

This simple query returns the status of automatic signal threshold selection.

```
ATHrhld?
    ATHrhld ON
    ATHrhld OFF
```





**AVDest** <arg>

Arguments: A, B, C

This single-argument command designates the spectrum analyzer display register used as the destination for ensemble average and minimum hold operations. The destination register cannot be changed while a MIN Hold or ensemble average operation is in progress.

```
AVDest A
AVDest B
AVDest C
```

**AVDest?**

Arguments: None

This simple query returns the spectrum analyzer display register currently selected as the destination for MIN Hold and ensemble average functions.

```
AVDest?
    AVDEST A
    AVDEST B
    AVDEST C
```

**AVG** <arg>

Arguments: ON,OFF

This single argument command turns the currently selected ensemble averaging mode on and off. Ensemble averages will terminate after the requested number of sweeps are averaged, but AVG OFF is used to terminate a continuous average. Ensemble averaging cannot be turned on if the analog display mode is active, or if there is a destination register conflict.

```
AVG ON
AVG OFF
```

**AVG?**

Arguments: None

This simple query returns the on/off status of the currently selected ensemble averaging mode.

```
AVG?
    AVG ON
    AVG OFF
```

**AVMode** <arg>

Arguments: MAX, MAXMin, MEAN, MIN

This single-argument command designates the ensemble average mode.

```
AVMode MAX
AVMode MAXMin
AVMode MEAN
AVMode MIN
```

**AVMode?**

Arguments: None

This simple query returns the currently selected ensemble averaging mode.

```
AVMode?
    AVMODE MAX
    AVMODE MAXMIN
    AVMODE MEAN
    AVMODE MIN
```

**AVNum** <arg>

Arguments: integer number in the range of 0 to 1024

This single-argument command designates the number of sweeps to be averaged by the currently selected ensemble averaging mode. If zero is specified, a continuous average is performed. The default is 16.

```
AVNum 128 (for example)
```

**AVNum?**

Arguments: None

This simple query returns the integer number of sweeps the current ensemble averaging mode will average.

```
AVNum?
    AVNUM 128 (for example)
```





**BWMode** <arg>

Arguments: ON, OFF, IDLE

This single-argument command turns the spectrum analyzer bandwidth measurement mode on and off. When bandwidth mode is turned on, marker modes are turned off if they were previously enabled. Bandwidth mode is not allowed if the spectrum analyzer is in analog display mode or Video Monitor (Option 10) mode. Bandwidth mode cannot be enabled for waveforms in the D-register if waterfall mode is enabled.

```
BWMode ON
BWMode OFF
BWMode IDLE
```

BWMode IDLE has the same effect as BWMode ON.

**BWMode?**

Arguments: None

This simple query returns the status of the bandwidth measurement mode.

```
BWMode?
    BWMODE ON
    BWMODE OFF
    BWMODE IDLE
```

**BWNum** <arg>

Arguments: Number in the range –1 to –70

This single argument command specifies the integer number of decibels (dB) below the signal peak at which the bandwidth measurement mode measures bandwidth. Units are not allowed; dBc units are assumed. Non-integer values are truncated.

```
BWNum -20 (for example)
```

**BWNum?**

Arguments: None

This simple query returns the integer number of decibels (dB) below the peak at which a signal's bandwidth will be measured by the spectrum analyzer's bandwidth measurement mode.

```
BWNum?
    BWNUM -20 (for example)
```

**BWResult?**

Arguments: None

This simple query returns the result of the most recent bandwidth measurement in Hertz (using the spectrum analyzer's bandwidth measurement feature). The result is updated at the end of the current sweep if bandwidth mode is not idle.

```
BWResult?
    BWRESULT 5.238E+3 (for example)
```

**CALSig** <arg>

Arguments: ON, OFF

This single-argument command turns the calibration signal on and off. The RF input signal is disconnected when the calibration signal is turned on.

```
CALSig ON
CALSig OFF
```

**CALSig?**

Arguments: None

This simple query returns the on/off status of the calibration signal.

```
CALSig?
    CALSIG OFF
    CALSIG ON
```

**CENsig**

Arguments: None

This is a command with no argument that sets the center frequency to the frequency of the primary marker.

```
CENsig
```

**CFSF** <arg>

Arguments: CENter, STArt

This single-argument command designates the displayed frequency as the center or start frequency. The indicated frequency may be adjusted, depending on the current center





frequency and frequency span, to ensure the resulting condition is permissible.

```
CFSF CENter
CFSF STArt
```

**CFSF?**

Arguments: None

This is a simple query whose response indicates whether the spectrum analyzer's on-screen frequency is the center or start frequency.

```
CFSF?
    CFSF CENTER
    CFSF START
```

**CLOck** <arg>

Arguments: ON, OFF

This single-argument command turns the date and time display on and off.

```
CLOck ON
CLOck OFF
```

**CLOck?**

Arguments: None

This simple query returns the status of the date and time display.

```
CLOck?
    CLOCK ON
    CLOCK OFF
```

**CLRKey**

Arguments: None

This is a command with no argument that clears the last key pressed so that only new key presses are reported by the KEY? query. After using the CLRKey command, KEY? returns NULL until a new key is pressed.

```
CLRKey
```

### CLRMenu

Arguments: None

This is a command with no argument that clears the menu defined with a DEFMenu command. This command clears the RAM space used by the User-Defined menu, takes the spectrum analyzer out of menu mode, and clears the last key pressed as reported by the KEY? query. With this command the spectrum analyzer always returns to the spectral display.

```
CLRMenu
```

### CMEas

Arguments: None

This is a command with no argument that causes the spectrum analyzer to perform a center measure. When a Frequency Counter is installed (2711 requires Option 02, Frequency Counter) use the COUnt? query to return the resulting counted value. Use the FREq? query to return the new center frequency when a Frequency Counter is not installed (2711 without Option 02, Frequency Counter).

```
CMEas
```

### CNBw <arg>

Arguments: frequency in the range 1 Hz to 1.8 GHz

This single-argument command specifies the bandwidth used by the carrier-to-noise (C/N) feature to perform a C/N measurement. Hertz are used if no units are appended.

```
CNBw 4.0 MHz (for example)
```

### CNBw?

Arguments: None

This simple query returns the noise bandwidth in Hertz used by the carrier-to-noise (C/N) feature to perform a C/N measurement.

```
CNBw?
    CNBW 4.0E+6 (for example)
```





### CNMode <arg>

Arguments: ON, OFF, IDLE

This single-argument command turns the carrier-to-noise (C/N) mode's on/off status. All marker modes are turned off when C/N mode is enabled.

```
CNMode ON
CNMode OFF
CNMode IDLE
```

CNMode IDLE has the same effect as CNMode ON.

### CNMode?

Arguments: None

This simple query returns the status of the carrier-to-noise mode.

```
CNMode?
    CNMODE OFF
    CNMODE ON
    CNMODE IDLE
```

The response CNMode IDLE is an indicator that there is no signal, the AM detector is not selected, the noise is too close to the spectrum analyzer noise floor, or analog mode is selected.

### CNResult?

Arguments: None

This simple query returns the result in decibels (dB) of the most recent carrier-to-noise (C/N) measurement performed by the spectrum analyzer's C/N feature. The measurement is updated at the end of the current sweep if C/N mode is enabled. CNMode must be ON to obtain valid results.

```
CNResult?
    CNRESULT -4.65E+1 (for example)
```

### CNTtrak<arg> (2711 requires Option 02, Frequency Counter)

Arguments: ON, OFF

This single-argument command enables and disables the frequency counter in signal track mode. The command sets counter resolution to 1 Hz when the counter is enabled.

```
CNTtrak ON
CNTtrack OFF
```

**CNTtrak?** (2711 requires Option 02, Frequency Counter)

Arguments: None

This is a simple query whose response indicates whether the frequency counter is on or off in signal track mode.

```
CNTtrak?
    CNTTRAK OFF
    CNTTRAK ON
```

**COUnt?** (2711 requires Option 02, Frequency Counter)

Arguments: None

This simple query returns the result (in Hertz) of the last frequency count performed as a result of the CMEas command.

```
COUnt?
    COUNT 55.250E+6 (for example)
```

**CREs** <arg> (2711 requires Option 02, Frequency Counter)

Arguments: 1 Hz, 1 kHz

This single-argument command designates the frequency counter resolution.

```
CREs 1 Hz
CREs 1 kHz
```

**CREs?** (2711 requires Option 02, Frequency Counter)

Arguments: None

This simple query returns the currently selected counter resolution. Values of 1 Hz and 1 kHz are possible.

```
CREs?
    CRES 1.E+0
    CRES 1.E+3
```

**CURve** <arg>

Arguments: Complex block of data described below.

This single-argument command enables a block of curve data to be sent to one of the spectrum analyzer's display registers. The data block represents the 512 horizontal points in a 2711 or 2712 waveform. The encoding of the data points (ASCII-encoded decimal, ASCII-encoded hexadecimal, or binary) is





determined by the current waveform preamble (see the WFMpre command). The register (A, B, C, or D) to which the data are sent is also determined by the preamble.

Data transferred to the spectrum analyzer can be previously returned waveforms or artificially generated curves. To ensure that the transferred data are not immediately overwritten, they should be transferred to a saved register (other methods are also possible, such as transferring to an active register in single sweep mode).

Figure 4-1 shows the format of curve data. The checksum definition ensures that the sum (modulo 256) of the data points plus point count plus checksum equals zero.

If you were to display the message on your controller screen (for instance, by printing the response to the CURve? query with HDR ON), the response would resemble this example:

binary responses always start with these characters

Binary: CURVE %☺☻&ÇwNc)*§>V...9¶;

hexidecimal responses always start with these characters

ASCII-encoded hex: CURVE #H02015E21B0F...E7B;

ASCII responses have no coding indicator or byte count characters

ASCII-encoded decimal: CURVE 94,233,7,182,...51,2,16;

Here are several more points worth noting about the various encodings:

- Binary encoding uses one byte per data point making it more compact and faster than the other techniques.

- Hexadecimal always uses two bytes per data point, thus it requires no delimiter to separate the points.
- Decimal uses a variable number of bytes (1 to 3) to encode each data point, and therefore requires a data point delimiter (the comma).
- The use of delimiters in decimal encoding makes this form compatible with many spread sheets and word processors. This enables you to create custom waveforms for later transmission to the spectrum analyzer, or to edit waveforms previously returned from the spectrum analyzer.

| CURVE<space><ind><$bc_{hi}$><$bc_{lo}$><$d_1$><$d_2$>...<$d_{512}$><checksum>; *where:* | |
|---|---|
| CURVE | Command header |
| space | Header delimiter |
| ind | Absent when ASCII-encoded decimal is used; equals #H when ASCII-encoded hexadecimal is used; equals the % sign when binary is used |
| $bc_{hi}$ | High order byte of the number of data points (always 512) plus one in the data block: absent in decimal, 00000010 in binary, 02 in hexadecimal |
| $bc_{lo}$ | Low order byte of the number of data points (always 512) plus one in the data block: absent in decimal, 00000001 in binary, 01 in hexadecimal |
| $d_1d_2...d_{512}$ | $1^{st}$ data point, $2^{nd}$ data point,...$512^{th}$ data point of the curve; each data point may be represented by one to four bytes depending on encoding:<br>• Binary: one byte<br>• Hexadecimal: two bytes representing the hexadecimal numerals 0-F<br>• Decimal: two-four bytes representing a comma delimiter plus one to three decimal numerals 0-9 |
| checksum | Absent for ASCII-encoded decimal; otherwise:<br>• 2's complement of {[$\Sigma d_i + bc_{hi} + bc_{lo}$] MOD 256} |

**Figure 4-1. Format of Curve Data.**





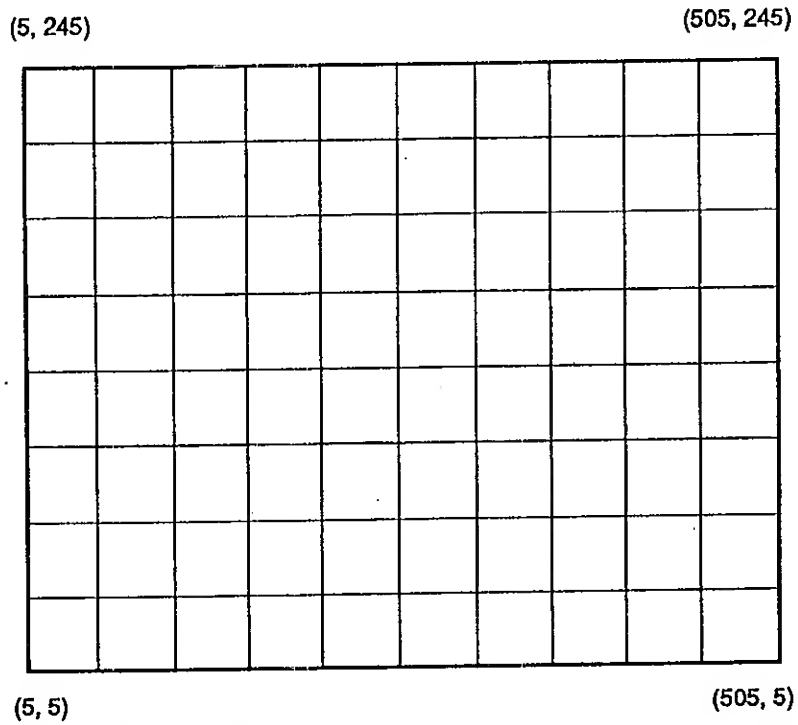



**Figure 4-2. Spectrum Analyzer Graticule Coordinates.**

The spectrum analyzer's graticule is represented by 500 intervals horizontally and 240 intervals vertically. The graticule corner coordinates are represented as shown in Figure 4-2.

Points along the horizontal axis are numbered from 0 to 511. The resulting graticule is represented by points 5 to 505. Values outside this range extend beyond the graticule area. Therefore, the sixth data point (5) along the horizontal axis crosses the left graticule line; the $505^{th}$ point crosses the right graticule line.

Vertically, the data are digitized into 256 values from 0 to 255. The resulting graticule is represented by points 5 to 245(Figure 4-2).. Values outside this range extend beyond the graticule area. Therefore, the sixth data point (5) along the vertical axis crosses the bottom graticule line; the $245^{th}$ point crosses the top graticule line. See Section 6, ***Programming***, for programming examples using CURve and CURve?.

**CURve? or CURve?** <arg>

Arguments: None, A, B, C, D

This is a query with either one or no arguments that returns a complex response representing the contents of a spectrum analyzer waveform display register.

```
CURve?
CURve? A
CURve? B
CURve? C
CURve? D
    CURVE 94,233,7,...151,2,16; (for example)
```

The format of the response is determined by the previous WFMpre command. If no argument is specified, the source register of the curve data is also determined by the previous WFMpre command. See the CURve command discussion for data formats and other details.

The CURve? response with HDR ON will resemble one of the forms shown in the CURve command discussion.

Data may be returned from an active or inactive register whether or not it is saved (the register always contains data even if it is not displayed). Contrast this function with the FILE command which returns a stored curve file whether or not it is currently in a register (in binary only, and not in curve data format).

See Section 6, ***Programming***, for programming examples using CURve?.

**DATe** <arg>

Arguments: Date in the form "DD-MON-YY"

This is a command with a string argument in the above format DD is a two-digit day of the month, YY is the last two digits of the year, and MON represents the first three letters of the month. This command sets the real-time clock date.

The DD and YY fields may be a single.digit If so, they are treated as if they had a leading digit of 0. The MON field may be any mixture of upper- and lower-case letters. Note that these elements are separated by hyphens (-), and the quotation marks (") must be present.

```
DATe "10-JAN-90"
```





**DATe?**

Arguments: None

This is a query that returns the current date in the format DD-MON-YY where DD is a two-digit day of the month, YY is the last two digits of the year, and MON represents the first three letters of the month.

```
DATe?
DATE "10-JAN-90"
```

**DATIme?**

Arguments: None

This is a query that returns the current date and current time in two comma-separated strings. The date is in the format DD-MON-YY where DD is a two-digit day of the month, YY is the last two digits of the year, and MON represents the first three letters of the month. The current time is in the format HH:MM:SS where HH is the hour, MM the minute, and SS the seconds.

```
DATIme?
DATIME "10-JAN-90","13:30:27"
```

**DEFMenu** <arg>

Arguments: L*n*:"user-defined string"

This is a command with arguments separated by commas in the form above. The L*n*: is called the link header, where *n* is a number between range 1 and 16 that defines a display line number. The link argument is a user-defined string that will appear on the specified display line. Only 32 characters of the string are used; excess characters are discarded. If fewer than 32 characters are contained in the link argument, the string is padded with spaces to a length of 32.

If the spectrum analyzer screen is displaying a spectral display or a built-in menu (at any level), this command places a new menu on the screen. If the instrument is already in a User-Defined menu, only the lines referenced in this command are replaced. Use CLRMenu to clear a User-Defined menu. The DEFMenu command also clears the last key press, but only if the User-Defined menu space is clear.

See Section 6, ***Programming***, for an example.

```
DEFMenu L0:"TEST MENU"
DEFMenu L0:"TEST MENU",L2:"TEST 1",L3:"TEST 2"
```

**DETector** <arg>

Arguments: AM, AMFm, FM, OFF

This single-argument command determines the type of signal detector used for the audio output. Depending on the argument, the AM, FM or both detector outputs are presented at the spectrum analyzer audio output. The audio output cannot be used in Video Monitor mode (Option 10).

```
DETector AM
DETector AMFm
DETector FM
DETector OFF
```

**DETector?**

Arguments: None

This simple query returns the current status of the audio source (whether the output of the AM detector, FM detector, neither, or both are being presented at the spectrum analyzer audio output).

```
DETector?
    DETECTOR AM
    DETECTOR AMFM
    DETECTOR FM
    DETECTOR OFF
```

**DIR?**

Arguments: None

This simple query returns a formatted system file directory listing similar to pressing [UTIL] [4] [6].

Each line in the listing (except the first and last) is formatted as in this example:

*filename, read/write enabled (R or W), size in bytes*

```
DIR?
    DIR " 12.88,        , 0
    TMPDBG, RW, 16380
      12.88,  ,         0
```





```
    DSET00, RW, 370
    SET0BU, RW, 370
        :
    UDP2, R , 160
    ";                  (for example)
```

Each line is separated by a line feed.

**DIScor** <arg>

Arguments: ON, OFF

This single-argument command enables and disables the spectrum analyzer's frequency corrections. When DIScor is ON (frequency corrections are disabled), the message "FREQ COR OFF" appears on the spectrum analyzer screen.

```
DIScor ON
DIScor OFF
```

**DIScor?**

Arguments: None

This simple query returns the current on/off status of the spectrum analyzer's frequency corrections. Note that DISCOR ON means the frequency corrections are off.

```
DIScor?
    DISCOR ON
    DISCOR OFF
```

**DLIne** <arg>

Arguments: ON, OFF

This single-argument command turns the display line feature on and off. DLINE cannot be turned on if the A-register is being used (waterfall mode, min hold, ensemble averaging, etc.). Attempting to do so generates an event 787, destination waveform conflict.

```
DLIne ON
DLIne OFF
```

**DLIne?**

Arguments: None

This simple query returns the current on/off status of the display line feature.

```
DLIne?
    DLINE ON
    DLINE OFF
```

**DLLimIt** <arg>

Arguments: OFF, OVEr, OVUNder, UNDer

This single-argument command controls the status of the display line limit detector. When the limit detector is not off, an SRQ and an event 895 are generated whenever the limit condition is exceeded. The following table shows the four arguments and their resulting condition.

| Argument | Condition |
|---|---|
| OVEr | Alarm when signal > display line |
| UNDer | Alarm when signal < display line |
| OVUNder | Alarm when signal > display line or when signal < threshold |
| OFF | No alarm |

```
DLLimit OFF
DLLimit OVEr
DLLimit OVUNder
DLLimit UNDer
```

**DLLimIt?**

Arguments: None

This simple query returns the current status of the display line limit detector.

```
DLLimit?
    DLLIMIT OFF
    DLLIMIT OVER
    DLLIMIT OVUNDER
    DLLIMIT UNDER
```





**DLValue** <arg>

Arguments: MARker, amplitude in the range –150 to +100 dBm

This single-argument command turns on the display line and sets its amplitude. A numeric argument sets the amplitude to the value of the argument. The units are the currently selected reference level units. However, the argument must be within a range of –150 dBm to +100 dBm, or an equivalent in alternate units. The MARKer argument sets the display line to the amplitude of the primary marker.

```
DLValue MARker
DLValue -30 (for example)
```

**DLValue?**

Arguments: None

This simple query returns the display line value. Units are the currently selected ref level units.

```
DLValue?
```

**DSRc** <arg>

Arguments: AM, EXTernal, FM, QP

This single-argument command designates the source of the signal displayed by the spectrum analyzer as the internal AM detector (normal display), internal FM detector (useful for FM deviation checks), or an EXTernal input.

The QP argument invokes the Quasi-Peak detector (2712 Option 12 only).

If FM or EXTernal are selected, the spectrum analyzer is placed in zero span mode and max/min signal acquisition is selected. FM and EXTernal are not allowed in DBUVM mode. EMC mode must be active before QP is selected; if not, Event 789 is declared and the display source is not changed.

If QP is selected, a "QP" precedes the vertical scale factor in the on-screen readout.

```
DSRc AM
DSRc EXTernal
DSRc FM
DSRc QP
```

### DSRc?

Arguments: None

This simple query returns the currently selected source of the signal displayed by the spectrum analyzer.

```
DSRc?
    DSRC AM
    DSRC FM
    DSRC EXTERNAL
    DSRC QP (2712 Option 12 only)
```

### EMC <arg> (Only available for 2712 Option 12, Quasi-Peak Detector)

Arguments: ON, OFF

This single-argument command turns EMC mode on and off.

```
EMC ON
EMC OFF
```

### EMC? (Only available for 2712 Option 12, Quasi-Peak Detector)

Arguments: None

This simple query returns the status of EMC mode.

```
EMC?
    EMC ON
    EMC OFF
```

### EOS <arg>

Arguments: ON, OFF

This single-argument command enables and disables the end-of-sweep SRQ. When EOS is ON, an end-of-sweep SRQ is normally generated at the end of each spectrum analyzer sweep. However, intermediate end-of-sweep SRQs are suppressed in the case of normalization, User Defined Programs, plots, signal searches, and ensemble averages until the process is complete. A single SRQ is then issued indicating "end-of-process."

For instance, if a 10-sweep ensemble average is compiled, an SRQ will not occur following each sweep. Rather, a single SRQ accompanied by event 882, ensemble average complete, occurs after the averaging process is finished.

```
EOS ON
EOS OFF
```





### EOS?

Arguments: None

This simple query returns the status of the end-of-sweep generator.

```
EOS?
    EOS ON
    EOS OFF
```

### ERAse <arg>

Arguments: Numerals from 2 to 39 except 9, 19 and 29

This single-argument command specifies a stored settings register to be erased. Registers 2 to 39 may be specified, except for registers 9, 19, and 29, which are invalid register numbers. If the settings are protected (locally or because PROTSET is ON), only the waveforms associated with the indicated settings are erased and an SRQ and event 839 are generated. If register 9, 19, or 29 is specified, an SRQ and Event 701 are generated.

```
ERAse 2
ERAse 24 (for example)
```

### ERr?

Arguments: None

This simple query returns an integer event code. ERr? is equivalent to EVEnt? and is preserved in the spectrum analyzer for consistency with the command sets of other instruments.

```
ERr?
    ERR 878 (for example)
```

### EVEnt?

Arguments: None

This simple query returns an integer event code. If RQS is ON, a serial poll must be performed following an SRQ and prior to sending EVEnt? to obtain the correct event code. If RQS is OFF, EVEnt? may be sent without a serial poll.

```
EVEnt?
    EVENT 878 (for example)
```

**FILE** <arg>

Arguments: "<filename>",<data block>

This is a complex single- or multiple-argument command that transfers a previously stored spectrum analyzer file from the controller to the spectrum analyzer. When used with only the <filename> argument, the command establishes the name of the file to be transferred to the controller by the next FILE? query. The "<filename>" must be surrounded by quotation marks ("). Tables 4-1 and 4-2 list the various user-alterable file types and allowable filenames.

The FILE command and FILE? query are intended to transfer files, through the mechanism of up-loading and subsequent down-loading, between different 2711 or 2712 spectrum analyzers. For instance, you might develop a User Definable Program (UDP) in one spectrum analyzer, transfer it to the controller using the FILE? query, and subsequently down-load it to a number of other spectrum analyzers using the FILE command. Be aware, however, that if files are transferred between spectrum analyzers with different installed options, uncertain results may occur.

The FILE command/query can also be used with a single spectrum analyzer to back up important settings files, or the reference normalizations, in preparation for changing the NVRAM battery.

When files are originally returned from the spectrum analyzer with the FILE? query, HDR is usually set ON so the ASCII strings FILE and <filename> precede the actual data and are stored as the first bytes of the disk file. It is then unnecessary to explicitly transmit the FILE header or the <filename> when restoring the file to the spectrum analyzer. Read the disk file into a string variable called FILEDAT$ as an example. The string variable will be of exactly the form needed to send the file to the spectrum analyzer as in this example.

```
FILE "<filename>",<data block>"
```

Simply transmit FILEDAT$ to the spectrum analyzer.





**Table 4-1. File Types.**

| File Type | Description |
|---|---|
| Settings | Each file saves the control settings for a particular register (A, B, C, D) in a numbered location (00-39, excluding 09, 19, & 29). The numbers correspond to the stored settings listed under [UTIL] [1]. |
| Curves | Each file saves the curve data from a particular register (A, B, C) in a numbered location (00-39, excluding 09, 19, & 29). The numbers correspond to the curves saved with the settings listed under [UTIL] [1]. D-register curves are never saved. |
| User-Definable Programs | Each file saves a keystroke command sequence representing a user-defined program in a numbered location (00 to 08); numbers correspond to the programs stored under [USER DEF]. |
| Antenna Tables | Each file saves an antenna table representing antenna data for a particular antenna in a numbered location (01 to 05); numbers correspond to the tables stored under [INPUT] [3] [9]. |
| Normalization | Normalization files save data generated by normalizing the spectrum analyzer, including reference normalizations. |

**Table 4-2. Valid File Names.**

| Curves | Settings | User-Definable Programs | Antenna Tables | Normalizations |
|---|---|---|---|---|
| nWFM00 | nSET00 | UDP00 | ACF1 | NORM |
| : | : | : | : | |
| nWFM09 | nSET09 | UDP08 | ACF5 | |
| n=A,B,C | n=A,B,C,D | | | |

Table 4-3. Miscellaneous Files.

| Name | Description |
|---|---|
| 12.88 | Version |
| SEARCH | Signal search configuration |
| SETUP | Instrument configuration |
| S_CENT | Centronics configuration |
| S_GPIB | GPIB configuration |
| S_PLOT | Plotter configuration |
| S_RTC | Real-time clock configuration |

Several other files of little use to the average user may be present, but should not be altered. These are listed in Table 4-3.

Other files of a temporary nature may be created by the spectrum analyzer for internal purposes. These files should not be altered.

If HDR was OFF when the file was returned, it is then necessary to precede the disk file with "FILE". In this case, transmit the following message:

```
FILE FILEDAT$
```

See Section 6, ***Programming***, for programming examples.

**FILE? or FILE?** <arg>

Arguments: None, "<filename>"

This is a simple query that returns a file stored in the spectrum analyzer to the controller. When used without an argument, FILE? returns the file specified by the previous FILE command. When used with a <filename> argument, FILE? returns the named file. The filename, when specified, must be in quotes ("). In either case the filename must match (including case) one of those listed in Table 4-2.

The file names in the 2711 and 2712 are established by their firmware. A directory of currently created files can be viewed by pressing [UTIL] [4] [6] or [UTIL] [5] [4] [1] [0]. Files are created within the spectrum analyzer's memory only as required. That is, a BSET03 settings file is only present when B-register settings have been stored previously in the third storage location.





DSET00 and SET0BU are two special files created automatically by the spectrum analyzer. They contain the D-register settings used when the spectrum analyzer was last turned off. They are listed as LAST POWER-DOWN under [UTIL] [1] [0]. SET0BU is a backup in case DSET00 becomes corrupted at the next power-up.

The FILE? query enables you to store a 2711 or 2712 file on disk for later restoration to the same or another 2711 or 2712. The file is in binary format, and the first bytes of the response are the ASCII character codes for <filename>. If HDR is ON before issuing the FILE? message, then the query and response have this format:

```
FILE? "<filename>"

    FILE "<filename>",<data block>
```

The response is in exactly the format needed to send a file to the spectrum analyzer. The general approach to file transfer is to read the response (including header and filename) into a string variable and write the variable to a disk file. The presence of FILE "<filename>" within the disk file makes it possible to restore the file to a 2711 or 2712 without having to explicitly send the FILE command or specify the spectrum analyzer file name.

Follow this sequence to store a 2711 or 712 file:

```
Send HDR ON to the 2711 (or 2712)

Send FILE? "<filename>" query to 2711 (or 2712)

Read response into string variable FILEDAT$

Write FILEDAT$ to disk file MYPROG
```

Use this sequence to restore the file:

```
Read file MYPROG to string variable FILEDAT$

Send FILEDAT$ to the 2711 (or 2712)
```

Be aware that all files except UDPs (and even most UDPs) occupy less than 5 kbytes of memory, but a UDP file can theoretically occupy up to 64 kbytes. Binary blocks are limited to 64 kbytes because of the 16-bit byte count.

See Section 6, ***Programming***, for programming examples.

**FINE** <arg>

Arguments: ON, OFF

This single-argument command selects 1 dB reference level steps when ON and 10 dB steps when OFF.

```
FINe ON
FINe OFF
```

**FINe?**

Arguments: None

This simple query returns the current on/off status of the reference level steps: ON equals 1 dB/step and OFF equals 10 dB/step

```
FINe?
    FINE ON
    FINE OFF
```

**FOFfset** <arg>

Arguments: frequency in the range –1000 GHz to +1000 GHz

This single-argument command turns on frequency offset mode (see FOMode) and sets the spectrum analyzer frequency offset value. A value of 0 turns off frequency offset mode. Frequency units may be appended; otherwise Hertz are assumed.

When enabled, the value of the frequency offset affects the display and any subsequent FREq, MFReq, STStop, SSBegin, and SSEnd commands.

```
FOFfset  5.15 GHz (for example)
```

**FOFfset?**

Arguments: None

This simple query returns the value of the frequency offset in Hertz. Zero is returned if the frequency offset is disabled.

```
FOFfset?
    FOFfset 0
    FOFfset 5.15E+9 (for example)
```





**FOMode** <arg>

Arguments: ON, OFF

This single-argument command turns frequency offset on or off.

When frequency offset is enabled the last offset frequency value is used (see FOFfset). When enabled, the value of the frequency offset affects the display and any subsequent FREq, MFReq, STStop, SSBegin, and SSEnd commands.

```
FOMode ON
FOMode OFF
```

**FOMode?**

Arguments: None

This simple query returns the current on/off status of the frequency offset mode.

```
FOMode?
    FOMODE OFF
    FOMODE ON
```

**FREq** <arg>

Arguments: frequency in the range –10 Hz to 1.8 GHz

This single-argument command sets the center or start frequency to the indicated value. Frequency units may be appended; otherwise Hertz are assumed. Interpretation as center or start frequency depends upon the setting of CFSF. If start frequency is selected, the span is checked and the start frequency may be adjusted to ensure the center frequency is never more than 1.8 GHz. The argument is offset by the FOFfset command if FOMode is enabled.

```
FREq 193.25 MHz (for example)
```

**FREq?**

Arguments: None

This simple query returns the currently selected center or start frequency.

```
FREq?
    FREQ 193.25E+6 (for example)
```

**GRAt <arg>**

Arguments: ON, OFF

This single-argument command turns the graticule illumination on and off.

```
GRAt OFF
GRAt ON
```

**GRAt?**

Arguments: None

This simple query returns the current on/off status of the graticule illumination.

```
GRAt?
    GRAT OFF
    GRAT ON
```

**GTL**

Arguments: None

This is a command with no arguments that removes the spectrum analyzer from the remote state and returns it to local mode. It is intended as the RS-232 equivalent of the GPIB universal GTL command (see ***Appendix A***).

```
GTL
```

**HDR <arg>**

Arguments: ON, OFF

This single-argument command turns the header on and off in a query response. When HDR is ON, a command header describing the nature of the response precedes the response proper. This also makes the response an executable command.

```
HDR OFF
HDR ON
```

The following table lists several queries and their potential responses with HDR ON and HDR OFF. Notice that in the case of most linked numerical arguments (but not those resulting from VRTdsp?, WAVfrm?, and WFMpre? queries), the link is turned off along with the command header, but not in the case of linked character arguments (see MFREq? and VIEw? in the table).





| HDR ON | HDR OFF |
|---|---|
| FREq?<br>FREQ 193.25E+6 | FREq?<br>193.25E+6 |
| GRAt?<br>GRAT ON | GRAt?<br>ON |
| MFREq? DELta<br>MFREQ DELTA:4.5E+6 | MFREq?DELta<br>4.5E+6 |
| VIEw? A<br>VIEW A:ON | VIEw? A<br>A:ON |

**HDR?**

Arguments: None

This simple query returns the current on/off status of the response header.

```
HDR?
    HDR OFF
    HDR ON
```

**HELp?**

Arguments: None

This simple query returns a list of all instrument-specific commands, including commands for all options whether present or not.

```
HELp?
    HELP ACQMODE,AQP,...WFMPRE,ZEROSP
```

**HRAmpl**

Arguments: None

This is a command with no argument that moves the primary marker from its current position to the peak of the next higher on-screen signal. If the marker is not enabled, the command enables a marker. If signal track is enabled, HRAmpl turns signal track off, enables the marker, and assigns the knob function to marker control. If there is no higher peak and SGErr is ON, an SRQ and event 896 are generated.

```
HRAmpl
```

### ID?

Arguments: None

This simple query returns the instrument identification, firmware version, and installed options.

```
ID?

    ID TEK/2712,V81.1,"VERSION 10.11.91

    FIRMWARE","300HZ,1,10,100KHZ,1MHZ

    REW FLTR","PHASE LOCK",300KHZ

    FILTER","VIDEOMONITOR","GPIB",

    "COUNTER","NVM 12.88";          (for example)
```

The items in quotes (") indicate the firmware version and options installed in the spectrum analyzer and may vary depending on how your spectrum analyzer is equipped.

### IMPCor <arg>

Arguments: Integer or floating point number

This single-argument command instructs the spectrum analyzer to scale its measurement results for 50 or 75 ohm input. Only the values 50 and 75 are valid. Numbers of 60 or below are rounded to 50; numbers above 60 are rounded to 75. Units are not allowed.

The spectrum analyzer has a 50 ohm input, but it can scale measurements to reflect values that would be measured if a 75 ohm instrument was used under the same conditions. See the *2711 Spectrum Analyzer User Manual* or *2712 Spectrum Analyzer User Manual* for a detailed description of the 50/75 ohm corrections.

```
IMPCor 75

IMPCor 50
```

### IMPCor?

Arguments: None

This simple query returns the input impedance value in ohms for which measurements are scaled.

```
IMPCor?

    IMPCOR 50

    IMPCOR 75
```





### INIT

Arguments: None

This is a command requiring no argument that places the spectrum analyzer in its power-up configuration. Factory default power-up settings are used unless user-defined power-up settings have been implemented.

```
INIT
```

### KEY <arg>

Arguments: various mnemonics as listed in Table 4-4.

This single-argument command simulates pressing a key on the spectrum analyzer front panel. The general form of the command resembles this example:

```
KEY <arg>
```

where <arg> is the mnemonic for the key press to be simulated. All permissible mnemonics are listed in Table 4-4. For instance, to simulate pressing the [INPUT Menu] key, you send the following message:

```
KEY INPutmenu  (for example)
```

To turn on the calibration signal using the KEY command, you can send this message:

```
KEY INPutmenu;KEY M9  (for example)
```

The KEY command is not as efficient as using a dedicated instrument-specific command, such as CALSig ON, to achieve the same result. It requires more time and memory space. Nevertheless, the command does provide an alternative if you experience difficulty implementing a dedicated command. In fact, you can perform all GPIB programming using only the KEY command. However, because of its increased memory requirement and decreased speed, use of the KEY command is discouraged as a general purpose GPIB programming technique. Note that there are no KEY commands for the [PLOT] and [POWER] keys.

Table 4-4. Arguments of the KEY Command.

| Mnemonic | Key |
|---|---|
| APplmenu | Application Menu |
| A | Display A |
| B | Display B |
| BS | Backspace key |
| C | Display C |
| D | Display D |
| CTRMeas | Center Measure |
| DEMMenu | Demodulator/Generator Menu |
| DIspmenu | Display Menu |
| FIne | Fine Ref Lvl steps |
| FREQAsgn | Assign Keypad to Frequency |
| FREQDown | Frequency Down |
| FREQUp | Frequency Up |
| INPutmenu | Input menu |
| KNOBRight | Turn knob right 1 step |
| KNOBLeft | Turn knob left 1 step |
| M0 | 0 key |
| M1 | 1 key |
| M2 | 2 key |
| M3 | 3 key |
| M4 | 4 key |
| M5 | 5 key |
| M6 | 6 key |
| M7 | 7 key |
| M8 | 8 key |
| M9 | 9 key |
| MAXHold | Max Hold |
| MAXSpan | Max Span |
| MKREnab | Marker On/Off/Delta |
| MKRLeft | Marker Left |
| MKRMenu | Marker/Frequency Menu |
| MKRPeak | Marker Peak |





Table 4-4. (Continued)

| Mnemonic | Key |
|---|---|
| MKRRight | Marker Right |
| PERIOD | Period |
| RDOut | Readout |
| REFAsgn | Assign keypad to Ref lvl |
| REFDown | Ref Lvl Down |
| REFUp | Ref Lvl Up |
| RESAuto | Auto ResBW |
| RESDown | ResBW Down |
| RESUp | ResBW Up |
| SAve | Save |
| SGLswp | Single Sweep |
| SPANAsgn | Assign keypad to Span |
| SPANDown | Span Down |
| SPANUp | Span Up |
| SWPAuto | Sweep Auto |
| SWPDown | Sweep Time Down |
| SWPMenu | Swp/Trig Menu |
| SWPUp | Sweep Time Up |
| TERMW | First Terminator |
| TERMX | Third Terminator |
| TERMY | Second Terminator |
| TERMZ | Fourth Terminator |
| USErdef | User Def Menu |
| UTilmenu | Utility Menu |
| VIDflt | Video Filter |
| VRTLIn | Lin Mode |
| VRTLOg | Log Mode |
| Zerospan | Zero Span |

### KEY?

Arguments: None

This simple query returns the identity of the last key pressed that has not been reported by a previous KEY? query. The name of the key uses the same syntax as the KEY command. If the key name returned is NULL, then no keys have been pressed since the last CLRMenu command or KEY? query has been executed. This query also clears the last key press, ensuring that the same key press is not reported more than one time.

```
KEY?

    KEY M1  (for example)

    KEY NULL  (for example)
```

**NOTE**

*NULL is returned if the [PLOT] key is the last key pressed.*

### LRAmpl

Arguments: None

This is a command with no argument that moves the primary marker from its current position to the peak of the next lower on-screen signal. If the marker is not enabled, the command turns on a marker. If signal track is enabled, LRAmpl turns it off, enables the marker, and assigns the knob function to marker control. If there is no lower peak and SGErr is ON, an SRQ and an event 896 are generated.

```
LRAmpl
```

### MAMpl? <arg>

Arguments: None, PRImary, SECond, DELta

This is a simple query with one or no arguments. It returns a linked response indicating the amplitude of the primary (<arg> = none or PRImary) or secondary (<arg> = SECond) marker, or their amplitude difference (<arg> = DELta). The applicable units are those currently selected for the reference level unit.

```
MAMpl?

    MAMPL PRIMARY:6.8  (for example)
```





```
MAMpl? SECond

    MAMPL SECOND:2.4  (for example)

MAMpl? DELta

    MAMPL DELTA:4.4  (for example)
```

### MARker <arg>

Arguments: ON, OFF, SINgle, DELta

This single-argument command turns markers on and off. Turning on a marker places the knob function in marker control and disables signal track mode, bandwidth measurement mode, noise measurement mode, and C/N measurement mode. The markers cannot be turned on in analog display or Video Monitor (Option 10) modes, or in waterfall mode unless the D register is enabled.

```
MARker ON       (Turns on primary marker)
MARker SINgle   (Turns on primary marker)
MARker DELta    (Turns on both markers)
MARker OFF      (Turn off all markers)
```

### MARker?

Arguments: None

This simple query returns the on/off status of the markers.

```
MARker?

    MARKER SINGLE

    MARKER DELTA

    MARKER OFF
```

### MEMory?

Arguments: None

This query returns two integer numbers separated by a comma. The first number represents the total amount of free NVRAM. The second number represents the largest contiguous block of free NVRAM. The values depend on the options installed and the number of waveforms, settings, programs, and other data stored in the spectrum analyzer's memory. Values are always multiples of 16.

```
MEMory?

    MEMORY 16464,3296  (for example)
```

### MEXchg

Arguments: None

This is a command that requires no argument. It interchanges the primary (stationary) and secondary (moveable) markers. If delta marker mode is not active, the command generates an SRQ and event code 825.

```
MEXchg
```

### MFReq <arg>

Arguments: number in the range —10 MHz to 1.8 GHz

This single argument command sets the frequency of the primary marker. Hertz are assumed unless units are appended. The over all range is –10 MHz to 1.8 GHz, but the value specified must be within the current spectrum analyzer on-screen frequency span or an SRQ and event code are generated. This command is not valid in zero span mode.

```
MFReq 193.25 MHz (for example)
```

### MFReq? <arg>

Arguments: `None, PRImary, SECond, DELta`

This is a simple query with one or no arguments. It returns a linked response indicating the frequency of the primary (`<arg> = none or PRImary`) or secondary (`<arg> = SECond`) marker, or their frequency difference (`<arg> = DELta`). The units are Hertz

```
MFReq?
    MFREQ PRIMARY:193.25E+6 (for example)
MFReq? SECond
    MFREQ SECOND:197.75E+6 (for example)
MFReq? DELta
    MFREQ DELTA:4.5E+6 (for example)
```

### MHDest <arg>

Arguments: `A, B,` or `C`

This single-argument command selects the MIN Hold destination waveform.

```
MHDest A
MHDest B
MHDest C
```





### MHDest?

Arguments: None

This simple query returns the MIN Hold destination waveform.

```
MHDest?
    MHDEST A
    MHDEST B
    MHDEST C
```

### MKTIme <arg>

Arguments: number in the range 0 to 20

This single argument command sets the time of the primary marker. The command is valid only in zero span mode. Seconds are assumed unless units are appended. The range is 0 to 20 seconds, but the value specified must be within the current spectrum analyzer on-screen time span or an SRQ and event code are generated.

```
MKTime 204 Msec (for example)
```

### MKTime? <arg>

Arguments: None, `PRImary, SECond, DELta`

This is a simple query with one or no arguments. It returns a linked response indicating the time of the primary (`<arg> = none or PRImary`) or secondary (`<arg> = SECond`) marker, or their time difference (`<arg> = DELta`). The units are seconds.

```
MKTime?
    MKTIME PRIMARY:4.67E-4 (for example)
MKTime? SECond
    MKTIME SECOND:8.98E-4 (for example)
MKTime? DELta
    MKTIME DELTA:4.31E-4 (for example)
```

### MLFtnxt

Arguments: None

This is a command requiring no argument. It moves the primary marker from its current position to the next signal peak to the left. If signal track is enabled, `MLFtnxt` turns signal track mode off, enables the primary marker, and assigns the knob function

to marker control. If SGErr is ON and a peak does not exist, an SRQ and event code are generated.

```
MLFtnxt
```

### MMAx

Arguments: None

This is a command requiring no argument. It moves the primary marker from its current position to the highest signal peak on screen. If signal track is enabled, MMAx turns signal track mode off, enables the primary marker, and assigns the knob function to marker control. If SGErr is ON and a higher peak does not exist, an SRQ and event code are generated.

```
MMAx
```

### MNHId <arg>

Arguments: OFF, ON

This single-argument command turns the minimum hold feature on and off. This command is not allowed under the following conditions:

- Analog mode is being used
- Waterfall mode is enabled
- The destination register is A and display line is on
- dBµV/m is enabled and destination register is the same as for min hold
- Ensemble averaging is enabled

```
MNHId ON
MNHId OFF
```

### MNHId?

Arguments: None

This simple query returns the on/off status of the minimum hold feature.

```
MNHId?
    MNHID ON
    MNHID OFF
```





### MPOs? <arg>

Arguments: None, PRImary, SECond, DELta

This is a query with one or no argument. It returns a linked integer response indicating the horizontal position of the primary (<arg> = none or PRImary) or secondary (<arg> = SECond) marker, or their horizontal difference (<arg> = DELta). See the CURve command for an explanation of screen coordinates.

```
MPOs?
    MPOS PRIMARY:356 (for example)
MPOs? SECond
    MPOS SECOND:233 (for example)
MPOs? DELta
    MPOS DELTA:123 (for example)
```

### MRGTnxt

Arguments: None

This is a command requiring no argument. It moves the primary marker from its current position to the next signal peak to the right. If signal track is enabled, MRGTnxt turns signal track mode off, enables the primary marker, and assigns the knob function to marker control. If SGErr is ON and a peak does not exist, an SRQ and event code are generated.

```
MRGTnxt
```

### MSGdlm <arg>

Arguments: Lf (line feed), Semicolon

This single-argument command that selects a semicolon or line feed character as a message delimiter.

```
MSGdlm Lf
MSGdlm Semicolon
```

### MSGdlm?

Arguments: None

This is a simple query whose response indicates the currently selected message delimiter.

```
MSGdlm?
    MSGDLM LF
    MSGDLM SEMICOLON
```

### MSTep

Arguments: None

This is a command that requires no argument. It is equivalent to turning the frequency/markers knob one click in the counterclockwise direction. The spectrum analyzer's response depends upon the currently selected knob function.

```
MSTep
```

### MTUNE <arg>

Arguments: Value in the range ±1.8 GHz

This single argument command changes the frequency of the primary marker by the indicated amount. Negative values indicate a decrease in frequency. Although the range is ±1.8 GHz, the value specified must position the new marker frequency within the spectrum analyzer's on-screen frequency span or an SRQ and event code are generated.

```
MTUNE 546 kHz (for example)
```

### MVPos? <arg>

Arguments: None, PRImary, SECond, DELta

This is a query with one or no argument. It returns a linked integer response indicating the vertical position of the primary (<arg> = none or PRImary) or secondary (<arg> = SECond) marker, or their vertical difference (<arg> = DELta). The 0-point is at the bottom of the screen. See the CURve command for a more complete explanation of screen coordinates.

```
MVPos?
    MVPOS PRIMARY:356 (for example)
MVPos? SECond
    MVPOS SECOND:233 (for example)
MVPos? DELta
    MVPOS DELTA:123 (for example)
```





### MXHld

Arguments: OFF, ON

This single-argument command turns the maximum hold feature on and off.

```
MXHld ON
MXHld OFF
```

### MXHld?

Arguments: None

This simple query returns the current on/off status of the maximum hold feature.

```
MXHld?
    MXHLD ON
    MXHLD OFF
```

### MXRlvl <arg>

Arguments: NOMinal, number in range –50 to –20

This single-argument command sets the required level of signal amplitude at the input to the spectrum analyzer's first mixer to produce full-screen (top graticule line) deflection in 2 dB steps. Odd values are rounded. Units are not allowed; the number is interpreted as dBm. NOMinal selects the factory default value of –30 dBm.

```
MXRlvl NOMinal
MXRlvl -24 (for example)
```

### MXRlvl?

Arguments: None

This simple query returns the signal amplitude required at the input to the spectrum analyzer's first mixer to deflect the display to the top graticule line.

```
MXRlvl?
    MXRLVL -30 (for example)
```

**MXSpn** <arg>

Arguments: OFF, ON

This single-argument command turns the maximum span mode on and off. The spectrum analyzer returns to the previously selected span/division when MAX Span is turned off.

```
MXSpn ON
MXSpn OFF
```

**MXSpn?**

Arguments: None

This simple query returns the current on/off status of the maximum span feature.

```
MXSpn?
    MXSPN ON
    MXSPN OFF
```

**NNBw** <arg>

Arguments: Number in the range 1 Hz to 1.8 GHz

This single-argument command sets the noise bandwidth for normalized noise mode measurements. Units may be appended; otherwise Hertz are assumed.

```
NNBw 4 MHz (for example)
```

**NNBw?**

Arguments: None

This simple query returns the noise bandwidth in Hertz to be used for normalized noise mode measurements.

```
NNBw?
    NNBW 4.0E+6 (for example)
```

**NNMode** <arg>

Arguments: OFF, ON, IDLE

This single-argument command turns the normalized noise measurement mode on and off. The command also sets TMODE to MARKER, ACQMODE to MAXMIN, and SGTRAK to OFF. The command cannot be used in analog mode, Video Monitor mode (Option 10), waterfall mode when the D-register is off, or in linear





display mode. Normalized noise mode measurements cannot be made on a saved waveform. The NNMode IDLE command is equivalent to NNMode ON. The IDLE response indicates when the mode is enabled but no signal is present to measure.

```
NNMode ON
NNMode OFF
NNMode IDLE
```

NNMode?

Arguments: None

This simple query returns the current status of the normalized noise measurement mode. IDLE indicates when the noise is too close to the spectrum analyzer noise floor, the AM detector is not enabled, MAX Span is active, the waveform is saved, or the spectrum analyzer is in analog display mode.

```
NNMode?
    NNMODE ON
    NNMODE OFF
    NNMODE IDLE
```

NNResult?

Arguments: None

This simple query returns the result of the most recent normalized noise measurement. The result is updated at the end of each sweep when the normalized noise measurement mode is enabled. The units are those selected as reference level units.

```
NNResult?
    NNRESULT -93.5 (for example)
```

NORM

Arguments: ALL, AMPlitude, FREquency, TG

This single-argument command instructs the spectrum analyzer to carry out the indicated normalizations. The TG argument is only valid when the Tracking Generator (Option 04) is installed.

```
NORM ALL          (all normalization except reference)
NORM AMPlitude    (amplitude normalizations)
NORM FREquency    (frequency normalizations)
NORM TG           (tracking generator normalizations)
```

### NORM?

Arguments: None

This simple query returns a formatted listing of the current normalization parameters.

The following parameter list shows the format of the response. Actual values displayed in each category vary depending on the instrument.

```
NORM?
    NORM "TEK 2711 (or 2712)
    CURRENT NORMALIZATION VALUES:
    ═══════════════════════════
    MISCELLANEOUS -- NORM VALUES
          :
    ──────────────────────────────────────────
    VCO NORMALIZATIONS
          :
    ──────────────────────────────────────────
    CF NORMALIZATIONS
          :
    ──────────────────────────────────────────
    REFERENCES
          :
    ──────────────────────────────────────────
    VERTICAL SCALE OFFSETS
          :
    ──────────────────────────────────────────
    LOG NORMALIZATIONS
          :
    ──────────────────────────────────────────
    FILTER SENSITIVITY
          :
    ──────────────────────────────────────────
    FILTER AMPLITUDES
          :
```





```
    ──────────────────────────────────────────
    VR FINE GAIN
          :
    ──────────────────────────────────────────
    VR GAIN STEPS
          :
    ──────────────────────────────────────────
    RF ATTEN, PREAMP & DET GAIN
          :
    ═══════════════════════════
    ";
```

### OBWMode <arg>

Arguments: `ON, OFF, IDLE`

This single-argument command specifies occupied bandwidth measurement mode. The occupied bandwidth measurement mode is enabled if on or idle.

```
OBWMode ON
OBWMode OFF
OBWMode IDLE
```

### OBWMode?

Arguments: None

This simple query returns the status of the occupied bandwidth measurement mode.

```
OBWMode?
    OBWMODE ON
    OBWMODE OFF
    OBWMODE IDLE
```

### OBWPcnt <arg>

Arguments: Numeral in the range of 1 to 99

This single-argument command specifies the percentage (1% to 99%) of occupied bandwidth for occupied bandwidth measurements.

```
OBWPcnt 40 (for example)
```

**OBWPcnt?**

Arguments: None

This simple query returns the occupied bandwidth percentage.

```
OBWPcnt?

    OBWPCNT 40 (for example)
```

**OBWResult?**

Arguments: None

This simple query returns the result of the most recent occupied bandwidth measurement (in Hertz).

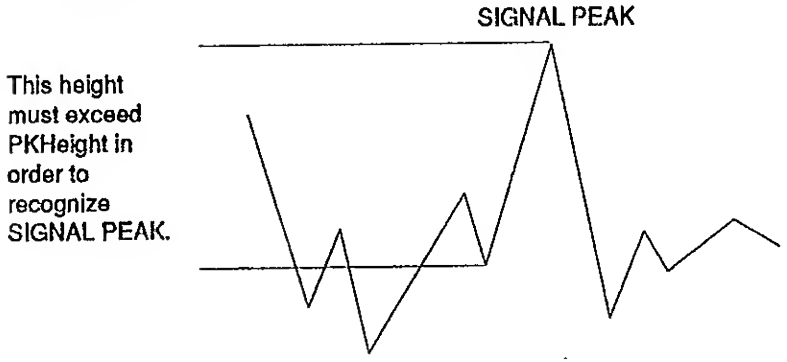
```
OBWResult?

    OBWRESULT 4.0E+6 (for example)
```

**PKHeight** <arg>

Arguments: Integer in the range 2 to 255

This single-argument command specifies how high a signal peak must be so it is recognized by the NEXT LOWER and NEXT HIGHER marker functions. The signal height is specified in.vertical display increments relative to the nearest local minimum in its skirts. 20 is the default value. Units are not allowed.




```
PKHeight 50 (for example)
```





**PKHeight?**

Arguments: None

This simple query returns an integer representing the signal height in vertical display increments that must exist for the peak to be recognized by the NEXT LOWER and NEXT HIGHER marker functions.

```
PKHeight?

    PKHEIGHT 20 (for example)
```

**PLLmode** <arg> (2712 Only)

Arguments: OFF, ON

This single-argument command enables or disables the 1st LO phase lock system. If the PLLmode is ON, the spectrum analyzer's 1st LO automatically phase locks for spans of 20 kHz/div or less.

```
PLLmode ON

PLLmode OFF
```

**PLLmode?** (2712 Only)

Arguments: None

This simple query returns the current on/off status of the 1st LO phase lock system.

```
PLLmode?

    PLLMODE ON

    PLLMODE OFF
```

**PLOT?**

Arguments: None

This simple query returns a complex response from the spectrum analyzer that provides screen plot information from for printing or plotting. The result of this command is similar to pressing the [PLOT] button on the front panel. The printer or plotter must speak the HPGL language or be compatible with Epson FX codes. The appropriate printer type must be specified locally or with the PTYpe command.

```
PLOT?

    <screen data array up to 61.1Kbyte long>
```

The data array can be up to 61.1 kbytes for Epson printers and up to 37 kbytes for HPGL plotters. PLOT? never produces a response header, even if HDR is ON. See section 6, ***Programming***, for programming examples.

**POFset** <arg>

Arguments: CENter, TOP

This single-argument command specifies whether to offset the result of the B,C MINUS A register arithmetic feature to the top or center of the display.

```
POFset CENter
POFset TOP
```

**POFset?**

Arguments: None

This simple query returns information indicating whether the result of the B,C MINUS A function is offset to the center or top of the spectrum analyzer display.

```
POFset?
    POFSET CENTER
    POFSET TOP
```

**PRDouts?**

Arguments: None

This simple query returns a list of the spectrum analyzer's on-screen readouts. There are up to 14 arguments depending on the status and mode of operation. These are the possible arguments:

- Title
- Center or start frequency
- Reference level
- Span/division
- Resolution bandwidth
- Attenuation or marker/delta frequency/normalized noise/carrier-to-noise/occupied bandwidth/frequency count
- Video filter or marker/delta amplitude/noise bandwidth/dB down for bandwidth mode





- Vertical scale
- Video line/TV channel number/average count/D line
- Single sweep mode/arm
- Tracking generator amplitude or amplitude offset
- CALIBRATOR or tracking generator frequency offset
- UNCAL or FREQ COR OFF
- Real time clock

The arguments are enclosed in quotation marks (") and separated by commas (,). The response ends in a semicolon (;). If an argument is missing, a null string ("") is returned. In principal, each argument can be up to 32 characters long (the string length is dimensioned for a maximum of 14 X 32 = 448 characters), but this length is never achieved in practice. The query does not return the spectrum analyzer's general purpose message line, GPIB status line, or user-defined "DISPLAY MESSAGE" line.

The following response example is returned after initializing the spectrum analyzer to the factory defaults. See section 6, ***Programming***, for programming examples.

```
PRDouts?
    PRDOUTS "","900MHZ","20.0DBM",
        "180MHZ/MAX","5MHZ RBW","ATTN
        50DB","VF WIDE","10DB/","","","","","","";
```

**PREamp** <arg>

Arguments: OFF, ON

This single-argument command turns the built-in preamplifier on and off.

```
PREamp ON
PREamp OFF
```

**PREamp?**

Arguments: None

This simple query returns the current on/off status of the built-in preamplifier.

```
PREamp?
    PREAMP ON
    PREAMP OFF
```

**PROTset** <arg>

Arguments: OFF, ON

This single-argument command turns stored settings protection on and off. Stored settings cannot be erased when PROTset is ON.

```
PROTset ON
PROTset OFF
```

**PROTset?**

Arguments: None

This simple query returns the current on/off status of the stored settings protection. Protected settings cannot be erased.

```
PROTset?
    PROTSET ON
    PROTSET OFF
```

**PSTep**

Arguments: None

This is a command that requires no argument. It is equivalent to turning the frequency/markers knob one click in the clockwise direction. The spectrum analyzer's response depends upon the currently selected knob function.

```
PSTep
```

**PTYpe** <arg>

Arguments: EPSON, HPGL2, HPGL4

This single-argument command specifies the type of printer or plotter encoding to use for screen data transferred from the spectrum analyzer to the controller in response to the PLOT? query.

```
PTYpe EPSON
PTYpe HPGL2
PTYpe HPGL4
```





**PTYpe?**

Arguments: None

This simple query returns the type of printer or plotter currently selected for use with the PLOT? query.

```
PTYpe?
    PTYPE EPSON
    PTYPE HPGL2
    PTYPE HPGL4.
```

**QPFilt** <arg> (Only available for 2712 Option 12, Quasi-Peak Detector)

Arguments: A, B, CD

This single-argument command selects the Quasi-Peak detector band for manual mode.

```
QPFilt A
QPFilt B
QPFilt CD
```

**QPFilt?** (Only available for 2712 Option 12, Quasi-Peak Detector)

Arguments: None

This simple query returns the Quasi-Peak detector band for manual mode.

```
QPFilt?
    QPFILT A
    QPFILT B
    QPFILT CD
```

**RECall** <arg>

Arguments: Integer 0 to 39 except 9, 19, and 29

This single-argument command instructs the spectrum analyzer to recall the stored settings in the location indicated by the argument. Integers between 0 and 39 (inclusive) are valid except for 9, 19, and 29.

```
RECall 0
RECall 24
```

**REDout** <arg>

Arguments: OFF, ON

This single-argument command turns the spectrum analyzer's on-screen readouts on and off.

```
REDout ON
REDout OFF
```

**REDout?**

Arguments: None

This simple query returns the on/off status of the spectrum analyzer's on-screen readouts.

```
REDout?
    REDOUT ON
    REDOUT OFF
```

**REFlvl** <arg>

Arguments: DEC, INC, ref level in the range –70 to +20 dBm

This single-argument command increases, decreases, or sets the reference level. If INC or DEC is the argument, the command increases or decreases the reference level by 1 dB or 10 dB depending on the FINe command or the local 1dB/10dB setting.

When a numeric argument is used, it must be within the range of –70 to +20 dBm (or equivalent in alternate units of DBM,DBMV, DBV, DBUV, DBUW, or DBUVM). If no units are used, the current reference level units are assumed. If units other than the current units are used, the value is converted to current units.

This command may alter the amount of RF attenuation if automatic RF attenuation is enabled. If LIN mode is active, the scale factor will be computed. All values are interpreted according to the current reference level offset and impedance correction.

```
REFlvl INC
REFlvl DEC
REFlvl 10 DBMV (for example)
```





**REFlvl?**

Arguments: None

This simple query returns the current reference level in the currently selected reference level units.

```
REFlvl?
    REFLVL -35.0 (for example)
```

**RESbw** <arg>

Arguments: INC, DEC, bandwidth of resolution bandwidth filter

This single-argument command increases, decreases, or selects the resolution bandwidth.

If INC or DEC is the argument, the command increases or decreases the resolution bandwidth to the next installed resolution bandwidth filter.

When a numeric argument is used the installed resolution bandwidth filter closest to the value is selected. Bandwidths available depend on the instrument type and installed options. This command disables automatic RES BW selection. If units are not attached, Hertz are assumed.

```
RESbw INC
RESbw DEC
RESbw 30 kHz (for example)
```

**RESbw?**

Arguments: None

This simple query returns the currently selected resolution bandwidth in Hertz.

```
RESbw?
    RESBW 3.0E+4 (for example)
```

**RFAtt** <arg>

Arguments: Number in the range 0 to 50

This single-argument command sets the RF attenuation to a fixed value between 0 and 50 dB in 2 dB steps. Values other than even integers are rounded. Units are not allowed.

```
RFAtt 34 (for example)
```

**RFAtt?**

Arguments: None

This simple query returns the current RF attenuation in decibels (dB) whether it is fixed or automatically selected.

```
RFAtt?
    RFATT 34 (for example)
```

**RLUnit** <arg>

Arguments: DBM, DBMV, DBV, DBUV, DBUW, DBUVM

This single-argument command specifies the indicated units for the reference level.

The DBUVM argument is not allowed under these conditions:

- Linear display mode
- DBUVM result is already saved
- Display source is the FM detector or external source
- There is a destination conflict with ensemble average, minimum hold, or display line

```
RLUnit DBM
RLUnit DBMV
RLUnit DBV
RLUnit DBUV
RLUnit DBUW
RLUnit DBUVM
```

**RLUnit?**

Arguments: None

This simple query returns the selected reference level units.

```
RLUnit?
    RLUNIT DBM
    RLUNIT DBMV
    RLUNIT DBV
    RLUNIT DBUV
    RLUNIT DBUW
    RLUNIT DBUVM
```





**ROFset** <arg>

Arguments: Value within the range ±100 dB

This single-argument command sets the reference level offset value. The offset value must range between –100 dB to +100 dB; units are not allowed.

```
ROFset -7.5 (for example)
```

**ROFset?**

Arguments: None

This simple query returns the current reference level offset. If the offset is disabled, the query returns 0. The units are decibels (dB).

```
ROFset?
    ROFSET -7.5 (for example)
```

**ROMode** <arg>

Arguments: OFF, ON

This single-argument command turns the reference level offset on and off.

```
ROMode ON
ROMode OFF
```

**ROMode?**

Arguments: None

This simple query returns the current reference level offset mode, either ON or OFF.

```
ROMode?
    ROMODE ON
    ROMODE OFF
```

**RQS** <arg>

Arguments: OFF, ON

This single-argument command enables and disables the generation of service requests (SRQ) by the spectrum analyzer. The user request is affected but the power-on SRQ is not.

```
RQS ON
RQS OFF
```

**RQS?**

Arguments: None

This simple query returns the spectrum analyzer's current on/off status of service request generation.

```
RQS?
    RQS ON
    RQS OFF (for example)
```

**RS232** <arg> (Requires Option 08, RS-232 Interface)

Arguments: (see following table)

| Arguments of RS232 and RS232? | Values |
|---|---|
| BAUd | 110, 150, 300, 600, 1200, 2400, 4800, 9600 |
| BITs | Number of data bits; 7 or 8 |
| ECHo | Echo mode; ON or OFF |
| EOL | Query termination; CR, LF, or CRLf |
| FLOw | Flow control; HARd, SOFt, NONe |
| PARity | Parity; ODD, EVEn, or NONe |
| VERbose | Verbose mode; ON or OFF |

This is a command that configures the RS-232 interface. The baud rate is set to the closest legal value when baud rates other than those listed are entered. One stop bit is selected unless the baud rate is 110, in which case two stop bits are selected.

```
RS232 BAUd:9600
RS232 VERbose:ON
```

**NOTE**

*Although it is syntactically correct to send more than one argument per command, it is dangerous to do so. Execution of these commands reprograms the interface. Also note that when* PARity: NONe *is selected,the spectrum analyzer does not add a parity bit on the data word, nor does it expect a parity bit on input. Some terminals transmit an 8-bit word with the 8th bit set to 0 when set to "7-bit, no parity." This can confuse the effort to ensure compatible settings between the spectrum analyzer and an external device. See the RS-232 program example in Section 6,* ***Programming.***





**RS232?** <arg> (Requires Option 08, RS-232 Interface)

Arguments: (see table for RS232 command )

This is a query that returns the current RS-232 parameter of the specified argument. If no argument is given, this command returns all RS-232 settings separated by commas (,).

```
RS232? FLOw
    RS232 FLOW:HARD (for example)
```

**RTIme** <arg>

Arguments: Time in the form "HH:MM:SS"

This command sets the time of the real-time clock. The argument is a string in the above format where HH is the hour in 24-hour format, MM is minutes, and SS is seconds. The seconds are set to zero regardless of the argument given.

Note that the elements are separated by colons (:). Quotation marks (") must be present. All fields must contain a value; a leading 0 is assumed if a single digit is used.

```
RTIme "13:30:00"
```

**RTIme?**

Arguments: None

This query returns the current time in the format HH:MM:SS where HH is the hour, MM is the minute, and SS is the seconds.

```
RTIme?
RTIME "13:30:27"
```

**SAVe** <arg>

Arguments: A:ON, A:OFF, B:ON, B:OFF, C:ON, C:OFF, and combinations of the above

This is a command with a single argument or multiple linked arguments. It saves and deletes waveforms located in NVRAM. ON saves the indicated display register to NVRAM. OFF deletes a saved display register from NVRAM.

For instance, the command SAVe A:ON saves the current contents of the A-register to NVRAM and halts the A-register from updating. SAVe A:OFF permits the A-register to be updated each sweep.

Single or multiple arguments can be used in a single command:

SAVe A:ON *(for example)*

SAVe C:OFF *(for example)*

SAVe A:ON,B:OFF *(for example)*

SAVe A:ON,B:OFF,C:OFF *(for example)*

Using the SAVe <link>:OFF command with the destination register for an ensemble average or minimum hold operation terminates these operations.

In addition, the DBUVM destination register cannot be turned off without first exiting DBUVM mode.

**SAVe?** <arg>

Arguments: None, A, B, C

This is a query with one or no argument that returns the storage state of the indicated register or all registers. If no argument is used, the state of the A, B, and C register is returned. If an argument is used, only the state of the indicated register is returned.

SAVe?

SAVE A:ON,B:OFF,C:OFF *(for example)*

SAVe? A

SAVE A:ON *(for example)*

SAVe? B

SAVE B:OFF *(for example)*

SAVe? C

SAVE C:OFF *(for example)*

**SET?**

Arguments: None

This simple query returns a group of command headers and arguments representing the current operating environment of the spectrum analyzer. The string of commands can be retained for transfer to the same or another 2711 or 2712 spectrum analyzer at a later time when it is desirable to reproduce the same operating environment. The SET? response enables you to replicate equipment setups. You should not modify the SET? response.





HDR status has no effect on the SET? query. Individual command headers are always returned and the group header (SET) is never returned. Each header and argument is separated by a semicolon (;) to ensure the response represents a functional message.

The SET? response is lengthy, but it is easy to interpret. A sample response follows that includes all 101 command headers that can be returned. Some of the headers depend on the options installed on your instrument and its particular configuration at the time of the SET? query. See the following examples.

- If the Tracking Generator (Option 04) or Video Monitor (Option 10) are not installed, all commands related to them will be absent (all commands beginning TG..., TVL..., and V... except VSYNC).
- If the spectrum analyzer is not in single sweep mode or if display line is not ON, the SIGSWP and DLVALUE commands will be missing. See the following example.

SET?

VIEW WATERFALL:OFF;RECALL 1;EMC OFF;DSRC
AM;VRTDSP LOG: 10;DETECTOR OFF;NNEW 1.0E+0;
NNMODE ON;CNEW 1.0E+0; CNMODE ON;HWNUM -3;
HWMODE ON;OBWPCNT 99;OBWMODE OFF;ACQMODE
MAXMIN; SPAN 180.E+6;MXSPN ON;ZEROSP OFF;
FOFFSET 0.000; FOMODE OFF;CFSF CENTER;FREQ
900.E+6;PLLMODE ON;DISCOR OFF;VMANITBL 1;VMDIST
3.0E+0;VMDEST C;VMMKRUNIT DBUVM;IMPCOR 50;
ROFSET 0.0;RLMODE OFF;RLUNIT DBM;WAIT;PREAMP
OFF;MXRLVL -30;REFLVL 20.0; RFATT 50;ARFATION;FINE
OFF;THRHLD -20.0;ATHRHLD ON; PKHEIGHT 20;DLVALUE
-20.0;DLINE ON;DLLIMIT OFF;RESBW 5.0E+6;ARES ON;
VIDFLT 5.0E+6;VFMODE AUTO; VFENAB OFF;MARKER
OFF;CRES1.E+3;SGTRAK OFF;AVDEST C;AVMODE MEAN;
AVNUM 16;AVG OFF;MNHLD OFF;MXHLD OFF;POFSET
CENTER;VIEW A:OFF,B:OFF,C:OFF,D:ON,MINUSA:OFF; STEP
3.600E+6;STPINC AUTO;TABLE 0;TGTRACK 0.000; TGTMODE
OFF;TGOOFFSET 0.0;TGOMODE OFF;TGMAN OFF;TGLEVEL

```
-48.0;TGENAB OFF;QPFILT CD;AQP OFF;TITLE "";  TIMODE
OFF;  TEXT "";REDOUT ON;GRAT OFF;PROTSET OFF;PTYPE
HPGL2;CALSIG OFF;MSGDIM SEMICOLON;HDR ON;EOS
OFF;SGERR OFF;RQS ON;TVLSID NTSC;VDMODE BROAD
CAST;VPOLARITY NEGATIVE;VSYNC POSITIVE;TVLINE 6;
TVLMODE     CONT;TIME 50.E-3;TIMMODE AUTO;TRIGGER
FRERUN;SIGSWP;TMODE FREQUENCY;VMONITOR ON;  TIME
50.E-3;SSBEGIN -10.000E+6;SSEND  1.036E+9;CLOCK ON;
```

### SGErr <arg>

Arguments: OFF, ON

This single-argument command enables and disables the generation of a service request (SRQ) when a marker function is unable to find a signal. SGErr ON enables SRQ generation for event 896.

```
SGErr ON
SGErrOFF
```

### SGErr?

Arguments: None

This simple query returns the on/off status of service request (SRQ) generation when a marker function cannot find the intended signal.

```
SGErr?
    SGERR ON
    SGERR OFF
```

### SGSrch

Arguments: None

This is a command that requires no argument. It instructs the spectrum analyzer to perform a signal search between the current BEGIN and END frequencies for all signals greater than the threshold (see THRhld). The BEGIN and END frequencies can be set locally or by using the SSBegin and SSEnd commands. Results of the search are returned by SSResult?

```
SGSrch
```





### SGTrak <arg>

Arguments: OFF, ON

This single-argument command enables and disables the signal track mode. The command does not work in zero span mode.

```
SGTrak ON
SGTrak OFF
```

### SGTrak?

Arguments: None

This simple query returns the on/off status of the signal track mode.

```
SGTrak?
    SGTRAK ON
    SGTRAK OFF
```

### SIGswp

Arguments: None

This is a command that requires no argument. It selects and arms the single sweep mode. The sweep does not actually occur until the trigger conditions for the currently selected trigger mode are satisfied. Any TRIGGER command cancels single sweep mode.

```
SIGswp
```

### SIGswp?

Arguments: None

This simple query returns the current status of the single sweep mode.

```
SIGswp?
    SIGSWP ON
    SIGSWP OFF
    SIGSWP ARM
```

**SPAn** <arg>

Arguments: 0, INC, DEC

2711 accepts values in the range 10 kHz to 180 MHz.

2712 accepts values in the range 1 kHz to 180 MHz.

This single-argument command increases, decreases, or sets the span/division.

When used with the INC or DEC argument, the command changes the span/division in the indicated direction in the normal 1-2-5 sequence.

The span/division is set to the indicated value for numeric arguments other than zero. If the value is out of range, the end point is substituted and an SRQ and event code are generated. If the 0 (zero) argument is used, zero span mode is activated.

```
SPAn INC
SPAn DEC
SPAn 0
SPAn 25 kHz (for example)
```

**SPAn?**

Arguments: None

This simple query returns the current span/div in Hertz.

```
SPAn?
    SPAN 2.5E+4 (for example)
```

**SSBegin** <arg>

Arguments: value in the range 9 kHz to 1.8 GHz

This single-argument command specifies the BEGIN frequency for the signal search mode. The BEGIN frequency must be less than the END frequency. Units may be appended; otherwise Hertz are assumed. The value is assumed to be offset by FOFfset if FOMode is ON.

```
SSBegin 54 MHz (for example)
```





**SSBegin?**

Arguments: None

This simple query returns the currently specified BEGIN frequency in Hertz for the signal search mode.

```
SSBegin?
    SSBEGIN 54.000e+6 (for example)
```

**SSEnd** <arg>

Arguments: Value in the range 9 kHz to 1.8 GHz

This single-argument command specifies the END frequency for the signal search mode. The END frequency must be greater than the BEGIN frequency. Units may be appended; otherwise Hertz are assumed. The value is assumed to be offset by FOFfset if FOMode is ON.

```
SSEnd 300 MHz (for example)
```

**SSEnd?**

Arguments: None

This simple query returns the currently specified END frequency in Hertz for the signal search mode.

```
SSEnd?
    SSEND 300.00E+6 (for example)
```

**SSResult?**

Arguments: None

This simple query returns the number of signals detected during a signal search operation, and lists the frequency and amplitude of each signal. Up to 50 frequency/amplitude pairs can be returned. The frequency and amplitude values are separated by commas (,), and the pairs of values are also delimited by commas.

The list begins with the lowest-frequency signal detected and proceeds to the highest. The amplitude units are those currently selected as reference level units; frequency is in Hertz.

Following is a typical example of the response when HDR is ON.

```
SSRESULT <N>,<freq1>,<ampl1>,...,<freqN>,<amplN>;
```

where:

<N> = number of signals detected (N ≤ 50)

<freq1>,...,<freqN> = frequency of 1st,...,Nth detected signal

<ampl1>,...,<amplN> = amplitude of 1st,...,Nth detected signal

If no signals are detected during the search, SSResult? returns zero (with HDR ON, the response is SSRESULT 0;). If the amplitude of a detected signal is off-screen, it is listed as 1.0E+6.

```
SSResult?
    SSRESULT 8,55.250E+6,7.3,...299.75E+6,-4.0;
        (for example)
```

See Section 6, ***Programming***, for programming examples.

### STByte?

Arguments: None

This simple query returns the GPIB serial poll response byte. This command is only useful for instruments equipped with the RS-232 interface. If the query is received by an instrument equipped with the GPIB interface the value 0 is always returned.

```
STByte?
    STBYTE 61 (for example)
```

### STEp <arg>

Arguments: CF, MARker, number in the range 1 Hz to 1.8 GHz

This single-argument command specifies the programmed frequency tuning increment. Specifying the increment also turns on the spectrum analyzer's programmed tuning mode. The CF argument selects the current center frequency as the increment, MARker selects the current marker frequency, and a numeric argument specifies the increment in Hertz. Units may be appended.

```
STEp CF
STEp MARker
STEp 30 kHz (for example)
```





### STEp?

Arguments: None

This simple query returns the currently specified programmed frequency tuning increment in Hertz.

```
STEp?
    STEP 3.0E+4 (for example)
```

### STOre <arg>

Arguments: Integers 2 to 39 except 9, 19, and 29

This single-argument command stores the spectrum analyzer control settings in the location designated by the argument. Locations 0 and 1 are reserved for the last power-down and factory default power-up settings, respectively. Locations 9, 19, and 29 are invalid.

```
STOre 2
STOre 24 (for example)
```

### STPInc <arg>

Arguments: AUTo, TABular, PROg

This single-argument command selects the tuning increment mode as automatic, tabular, or programmed. Refer to the STEp command to set the programmed increment.

```
STPinc AUTo
STPinc TABular
STPinc PROg
```

### STPInc?

Arguments: None

This simple query returns the currently selected tuning increment mode.

```
STPinc?
    STPINC AUTO
    STPINC TABULAR
    STPINC PROG
```

**STStop** <arg>

Arguments: MARker or pair of values in range –10 MHz to 1.8 GHz

This single- or double-argument command sets the START and STOP frequencies of the spectrum analyzer display. If MARker is the argument, the start and stop frequencies are set to the current marker frequencies (delta marker mode must be enabled).

If the argument is a pair of numbers, the first number specifies the START frequency and the second specifies the STOP frequency. Units may be appended; otherwise Hertz are assumed.

The second number must be at least 10 kHz greater than the first to satisfy the spectrum analyzer's span requirements. If the second number is greater than the first, but by less than 10 kHz, the STOP frequency is set to the START frequency plus 10 kHz.

The STOP frequency may be set lower than the START frequency. Under these conditions the spectrum analyzer is tuned to the lower frequency and zero span mode is activated.

```
STStop MARker
STStop 192 MHz,198 MHz  (for example)
```

**TABle** <arg>

Arguments: Integer number in the range 0 to 9

This single-argument command selects the tuning table to be used when tabular tuning increment mode is active. This command does not turn on tabular tuning; it only selects the table. Only tables 0 to 7 are filled. The number of an empty table can be entered but an SRQ and event code will be generated.

```
TABle 0
   :
TABle 7
```

**TABle?**

Arguments: None

This simple query returns the selected tabular tuning table.

```
TABle?
   TABLE 3  (for example)
```





**TAMpl?**

Arguments: None

This simple query returns the amplitude of the signal being tracked. The value is updated at the end of each sweep when signal track mode is enabled. Otherwise the amplitude of the signal last tracked is returned. The units are those currently selected as reference level units.

```
TAMpl?
   TAMPL -34.0  (for example)
```

**TEXt** <arg>

Arguments: String of up to 32 ASCII characters

This single-argument command displays a message on line 8 of the spectrum analyzer screen (line 9 if title mode is active). The message is the argument of the command (up to 32 characters). Quotation marks (") must be used. Only upper-case characters can be displayed, although lower case characters may be sent. Transmit a null string (TEXt "") to erase the message.

```
TEXt "MY MESSAGE"  (for example)
```

**TEXt?**

Arguments: None

This simple query returns the current contents of the message buffer in the spectrum analyzer. If lower-case letters were originally sent lower case letters are returned, even though the on-screen message is upper case.

```
TEXt?
   TEXT "MY MESSAGE"  (for example)
```

**TFReq?**

Arguments: None

This simple query returns the frequency (in Hertz) of the signal being tracked. The value is updated at the end of each sweep when signal track mode is enabled. Otherwise the amplitude of the signal last tracked is returned.

```
TFReq?
   TFREQ 101.36E+6  (for example)
```

**TGEnab** <arg> (Requires Option 04, Tracking Generator)

Arguments: OFF, ON

This single-argument command turns the optional tracking generator on and off. The tracking generator must be installed.

```
TGEnab ON
TGEnab OFF
```

**TGEnab?** (Requires Option 04, Tracking Generator)

Arguments: None

This simple query returns the current on/off status of the optional tracking generator.

```
TGEnab?
    TGENAB ON
    TGENAB OFF
```

**TGLevel** <arg> (Requires Option 04, Tracking Generator)

Arguments: Number in range –48 dBm to 0 dBm, or equivalent

This single-argument command sets the output amplitude of the optional tracking generator. The value specified may be in DBM, DBMV, DBV, DBUV, or DBUW (DBUVM defaults to DBUV), but its equivalent value must range between –48 dBm to 0 dBm. The level can be changed in 0.1 dB steps. If units are not appended, the current reference level units are assumed.

```
TGLevel 1.2 DBMV (for example)
```

**TGLevel?** (Requires Option 04, Tracking Generator)

Arguments: None

This simple query returns the currently specified output amplitude of the optional tracking generator. The units are those currently selected as the reference level units.

```
TGLevel?
    TGLEVEL 1.2 (for example)
```





**TGMan** <arg> (Requires Option 04, Tracking Generator)

Arguments: OFF, ON

This single-argument command enables and disables fine manual adjustment of the optional tracking generator output amplitude. In manual mode, the spectrum analyzer's TRIGGER LEVEL knob adjusts the tracking generator output amplitude about the level established with TGLevel. When TGMan is ON, the level returned by TGLevel? may differ slightly from the actual level.

```
TGMan ON
TGMan OFF
```

**TGMan?** (Requires Option 04, Tracking Generator)

Arguments: None

This simple query returns the on/off status of manual tracking generator amplitude control.

```
TGMan?
    TGMAN ON
    TGMAN OFF
```

**TGOMode** <arg> (Requires Option 04, Tracking Generator)

Arguments: OFF, ON

This single-argument command turns the output level offset of the optional tracking generator on and off.

```
TGOMode ON
TGOMode OFF
```

**TGOMode?** (Requires Option 04, Tracking Generator)

Arguments: None

This simple query returns the current on/off status of the tracking generator output offset.

```
TGOMode?
    TGOMODE ON
    TGOMODE OFF
```

**TGOOffset** <arg> (Requires Option 04, Tracking Generator)

Arguments: Value within the ±100 dB range

This single-argument command specifies the output level offset of the optional tracking generator. The offset can be between –100 dB to +100 dB. Units are not allowed. A non-zero argument turns offset mode on and a 0 argument turns the offset mode off.

```
TGOOffset -10.5 (for example)
```

**TGOOffset?** (Requires Option 04, Tracking Generator)

Arguments: None

This simple query returns the currently specified output level offset of the optional tracking generator in decibels (dB).

```
TGOOffset?
    TGOOFFSET -10.5 (for example)
```

**TGTMode** <arg> (Requires Option 04, Tracking Generator)

Arguments: OFF, ON

This single-argument command turns the frequency offset of the optional tracking generator on and off.

```
TGTMode ON
TGTMode OFF
```

**TGTMode?** (Requires Option 04, Tracking Generator)

Arguments: None

This simple query returns the current on/off status of the optional tracking generator's tracking mode.

```
TGTMode?
    TGTMODE ON
    TGTMODE OFF
```





**TGTRack** <arg> (Requires Option 04, Tracking Generator)

Arguments: Value within the range –5.01 kHz to +60 kHz

This single-argument command specifies the tracking adjustment of the optional tracking generator. Units may be appended; otherwise Hertz are assumed. A non-zero value turns on the tracking generator's tracking mode; a zero value turns the tracking mode off.

```
TGTRack 9.7 kHz (for example)
```

**TGTRack?** (Requires Option 04, Tracking Generator)

Arguments: None

This simple query returns the currently specified tracking value of the optional tracking generator in Hertz.

```
TGTRack?
    TGTRACK 9.7E+3 (for example)
```

**THRhld** <arg>

Arguments: Number within the range –174 to 20 dBm

This single-argument command specifies the value of the threshold above which the spectrum analyzer automatically detects signals. Units of DBM, DBMV, DBV, DBµV, DBUW, and DBUV/M can be used, but the equivalent value must be within the range –174 dBm to +20 dBm. If units are not supplied, the current reference level units are assumed. This command also turns off the automatic threshold selection mode.

```
THRhld -10 DBMV (for example)
```

**THRhld?**

Arguments: None

This simple query returns the current threshold value (fixed or automatically selected). The units are the currently selected reference level units.

```
THRhld?
    THRHLD -10.0 (for example)
```

**TIME** <arg>

Arguments: INC, DEC, value in the range 1 microsec to 2 sec

**NOTE**

*The TIME argument's range is limited to 100 microsec to 2 sec when any of the Display Storage registers (A, B, C, D) are active.*

This single-argument command increases, decreases, or sets the sweep speed. This command also turns off the spectrum analyzer's automatic sweep speed selection. Units of NS, US, MS, or S may be appended; otherwise seconds are assumed.

If the INC or DEC argument is used the sweep speed is increased or decreased in the normal 1-2-5 sequence. Sweep times that are not in the sequence are increased to the next valid setting.

Numeric arguments must be within the range of 1 microsecond to 2 seconds. Sweep times less than 100 microseconds are not permitted in display storage mode.

```
TIMe INC
TIMe DEC
TIMe 25 US (for example)
```

**TIMe?**

Arguments: None

This simple query returns the currently selected sweep speed. Units are in seconds.

```
TIMe?
    TIME 25.E-6 (for example)
```

**TIMMode** <arg>

Arguments: AUTo, MANual, FIXed

This single-argument command enables (AUTo) and disables (FIXed) automatic sweep speed selection. This command also enables manual sweep positioning (MANual) using the spectrum analyzer's LEVEL control. When in the FIXed mode the sweep speed is controlled locally or with the TIME command.

```
TIMMode AUTo
TIMMode FIXed
TIMMode MANual
```





**TIMMode?**

Arguments: None

This simple query returns the current time base mode.

```
TIMMode?
    TIMMODE AUTO
    TIMMODE FIXED
    TIMMODE MANUAL
```

**TITLe** <arg>

Arguments: String of up to 32 ASCII characters

This single-argument command displays a title on line 1 of the spectrum analyzer screen. The title is the argument of the command which may be up to 32 characters long. Quotation marks (") must be used. Only upper-case characters can be displayed. A null string (TITle "") is transmitted to erase the title. Use the TTLMode command to turn the title on or off.

```
TITle "SCREEN 1" (for example)
```

**TITLe?**

Arguments: None

This simple query returns the spectrum analyzer screen title if one currently exists. If the title was sent in lower-case letters, the returned string will be lower-case even though the title is displayed in upper-case letters on the spectrum analyzer screen.

```
TITle?
    TITLE "SCREEN 1" (for example)
```

**TMOde** <arg>

Arguments: FREquency, MARker, TG, VIDline

This single-argument command selects the frequency/marker knob function.

| Argument | Function |
|---|---|
| FREquency | Adjust start or center frequency |
| MARker | Adjust marker frequency |
| TG | Adjust tracking generator tracking |
| VIDline | Select video line number if knob selectable, TV line triggering is enabled |

```
TMOde FREquency
TMOde MARker
TMOde TG
TMOde VID line
```

### TMOde?

Arguments: None

This simple query returns the currently selected function of the frequency/markers knob.

```
TMOde?
    TMODE FREQUENCY
    TMODE MARKER
    TMODE TG
    TMODE VIDLINE
```

### TOPsig

Arguments: None

This is a command that requires no argument. It instructs the spectrum analyzer to change the reference level to the amplitude of the primary marker. A marker must be enabled.

```
TOPsig
```

### TRIgger <arg>

Arguments: `EXTernal, FRErun, INTernal, LINe, TVField, TVLine`

This single-argument command selects the trigger type. `TVLine` also sets the knob function to VIDLINE if knob-selectable TV line mode is enabled.

```
TRIgger EXTernal
TRIgger FRErun
TRIgger INTernal
TRIgger LINe
TRIgger TVField
TRIgger TVLine
```





### TRIgger?

Arguments: None

This simple query returns the selected spectrum analyzer trigger type.

```
TRIgger?
    TRIGGER FRERUN
    TRIGGER EXTERNAL
    TRIGGER INTERNAL
    TRIGGER LINE
    TRIGGER TVFIELD
    TRIGGER TVLINE
```

### TTLMode <arg>

Arguments: `OFF, ON`

This single-argument command turns the spectrum analyzer screen title on and off.

```
TTLMode ON
TTLMode OFF
```

### TTLMode?

Arguments: None

This simple query indicates whether the spectrum analyzer's screen title is being displayed (`ON`) or is not displayed (`OFF`).

```
TTLMode?
    TTLMODE ON
    TTLMODE OFF
```

### TUNe <arg>

Arguments: Frequency in the range –1.8 GHz to +1.8 GHz

This single-argument command changes the start or center frequency by the amount of the argument. The resultant frequency must remain within the range of –10 MHz to 1.8 GHz. Units may be appended; otherwise Hertz are assumed.

```
TUNe 10.8 MHz (for example)
```

**TVLine<arg>** (Requires Option 10, Video Monitor)

Arguments: Integer in the range 1 to 1024

This single-argument command specifies the number of the TV line to which the spectrum analyzer is to trigger when programmed TV line triggering is enabled. This command also turns off Video Monitor mode if it is enabled. The minimum argument is 1. The maximum value depends on the TV line standard; 525 for NTSC, 625 for PAL and SECAM and 1024 for OPEN. This command also sets `TRIgger` to `TVLine` and enables the programmed mode.

```
TVLine 17 (for example)
```

**TVLine?** (Requires Option 10, Video Monitor)

Arguments: None

This simple query returns the (integer) TV line number to which the spectrum analyzer is to trigger when TV line trigger mode is selected.

```
TVLine?
    TVLINE 17 (for example)
```

**TVLMode** <arg>

Arguments: `CONT, KNOB, PROg`

This single-argument command designates the specific TV line trigger mode when the Option 10,Video Monitor is installed, and enables TVLine trigger mode in all spectrum analyzers. This command turns the Video Monitor mode off if it is enabled. The arguments and their functions are described in the following table.

| Argument | Function |
|---|---|
| CONT | Trigger on every line |
| KNOB | Trigger line number selected with the frequency/markers knob |
| PROg | Trigger line number entered locally or with the TVLine command |

```
TVLMode KNOB (for example)
```





**TVLMode?** (Requires Option 10, Video Monitor)

Arguments: None

This simple query returns the currently selected TV line trigger mode.

```
TVLMode?
    TVLMODE CONT
    TVLMODE KNOB
    TVLMODE PROG
```

**TVLStd** <arg>

Arguments: `NTSC, OPEN, PAL, SECAM`

This single-argument command designates the TV standard when using the TV line trigger mode. This command turns the Video Monitor mode (Option 10) off if it is enabled, and sets the trigger mode to TV Line in all spectrum analyzers. The maximum value of the `TVLine` command's argument is influenced by the TV standard.

```
TVSTd NTSC
TVLStd OPEN
TVLStd PAL
TVLStd SECAM
```

**TVLStd?**

Arguments: None

This simple query returns the currently selected TV standard.

```
TVLStd?
    TVLSTD NTSC
    TVLSTD OPEN
    TVLSTD PAL
    TVLSTD  SECAM
```

**VDMode** <arg> (Requires Option 10, Video Monitor)

Arguments: `BROadcast, SATellite`

This single-argument command designates the type of detection to be used in the Video Monitor mode. `BROadcast` selects AM detection for use with broadcast television; `SATellite` selects FM detection for use with satellite transponders.

```
VDMode BROadcast
VDMode SATellite
```

**VDMode?** (Requires Option 10, Video Monitor)

Arguments: None

This simple query returns the currently selected detection for Video Monitor mode.

```
VDMode?
    VDMODE BROADCAST
    VDMODE SATELLITE
```

**VFEnab** <arg>

Arguments: `OFF, ON`

This single-argument command turns the video filter on and off.

```
VFEnab ON
VFEnab OFF
```

**VFEnab?**

Arguments: None

This simple query returns the current on/off status of the video filter.

```
VFEnab?
    VFENAB ON
    VFENAB OFF
```

**VFMode** <arg>

Arguments: `AUTo, FIXed`

This single-argument command enables (`AUTo`) and disables (`FIXed`) automatic selection of the video filter bandwidth. When fixed mode is first entered, the automatically selected video filter bandwidth is made the current fixed filter bandwidth. A new fixed filter bandwidth can then be selected locally or by using the `VIDflt` command.

```
VFMode AUTo
VFMode FIXed
```





**VFMode?**

Arguments: None

This is a simple query whose response indicates whether the video filter bandwidth is fixed or automatically selected by the spectrum analyzer.

```
VFMode?
    VFMODE AUTO
    VFMODE FIXED
```

**VIDflt** <arg>

Arguments: `OFF, ON`, floating point number

This single-argument command turns the video filter on or off, or specifies the filter bandwidth.

The `ON` and `OFF` arguments enable and disable the currently selected video filter in the same way as the `VFEnab` command. A numeric argument is used to specify a particular video filter bandwidth and turn on the video filter. Units may be appended; otherwise Hertz are assumed. The video filter bandwidths follow a 1-3 sequence. The video filter bandwidth closest to the specified filter width is selected.

```
VIDflt ON
VIDflt OFF
VIDflt 30 kHz (for example)
```

**VIDflt?**

Arguments: None

This simple query returns the currently selected video filter bandwidth in Hertz whether or not the filter is enabled.

```
VIDflt?
    VIDFLT 3.0E+4 (for example)
```

**VIEw** <arg>

Arguments: `A:ON, A:OFF, B:ON, B:OFF, C:ON, C:OFF, D:ON, D:OFF, Minusa:ON, Minusa: OFF, Waterfall:ON, Waterfall:OFF`, combinations of the above

This is a command with single- or multiple-linked arguments that enables and disables digital display mode in the indicated

register. For instance, VIEw A:ON turns on the digital display in the A-register; VIEw A:OFF turns off the A-register. Single or multiple arguments can be used in a single command as in this example:

VIEw B:ON *(for example)*

VIEw A:ON,B:OFF *(for example)*

VIEw A:ON,B:OFF,C:OFF *(for example)*

VIEw Waterfall:ON,A:OFF *(for example)*

Multiple arguments must be separated with commas (,).

### VIEw?

Arguments: None, A, B, C, D, Minusa, Waterfall

This is a query with one or no argument that returns the on/off status of the indicated register, or all storage registers. If no argument is used, the status of the A, B, C, and D registers, the waterfall mode, and B,C minus A display mode are returned. Only the state of the indicated register is returned when an argument is used.

VIEw?

VIEW Waterfall:OFF,A:ON,B:OFF,C:OFF,

D:ON,Minusa:OFF *(for example)*

VIEw? B

VIEW B:OFF *(for example)*

### VMAnttbl <arg>

Arguments: Integer in the range 1 to 5

This single-argument command designates by number the antenna table to be used for DBUVM measurements. Units are not allowed.

VMAnttbl 3 *(for example)*

### VMAnttbl?

Arguments: None

This simple query returns the number of the antenna table currently selected for use when making DBUVM measurements.

VMAnttbl?

VMANTTBL 3 *(for example)*





### VMDEst <arg>

Arguments: A, B, C

This single-argument command designates the spectrum analyzer display register used as the destination for DBUVM measurements.

VMDEst B *(for example)*

### VMDEst?

Arguments: None

This simple query returns the currently selected destination register for DBUVM measurements.

VMDEst?

VMDEST B *(for example)*

### VMDIst <arg>

Arguments: Floating point number

This single-argument command specifies the source-antenna distance at which a DBUVM measurement is actually performed. Distance may be entered in feet (FT), meters (M), kilometers (KM), or miles (MI), but the spectrum analyzer converts all values to meters or kilometers.

VMDIst 3 M *(for example)*

### VMDIst?

Arguments: None

This simple query returns the currently specified source-antenna distance for DBUVM measurements. Units are meters.

VMDIst?

VMDIST 3.0 *(for example)*

### VMMkrunit <arg>

Arguments: DBUVM, VM

This single-argument command specifies the amplitude units for marker readouts in DBUVM mode as DBUVM or volts/m.

VMMkrunit DBUVM

VMMkrunit VM

**VMMkrunit?**

Arguments: None

This simple query returns the currently selected units for marker readouts in the DBUVM mode.

```
VMMkrunit?
    VMMKRUNIT DBUVM
    VMMKRUNIT VM
```

**VMOnitor** <arg> (Requires Option 10, Video Monitor)

Arguments: `OFF, ON`

This single-argument command turns the Video Monitor on and off.

```
VMOnitor ON
VMOnitor OFF
```

**VMOnitor?** (Requires Option 10, Video Monitor)

Arguments: None

This simple query returns the current on/off status of the Video Monitor mode.

```
VMOnitor?
    VMONITOR ON
    VMONITOR OFF
```

**VPOlarity** <arg> (Requires Option 10, Video Monitor)

Arguments: `NEGative, POSitive`

This single-argument command specifies the polarity of video signals to be received with the Video Monitor mode.

```
VPOlarity NEGative
VPOlarity POSitive
```

**VPOlarity?** (Requires Option 10, Video Monitor)

Arguments: None

This simple query returns the currently specified video signal polarity.

```
VPOlarity?
    VPOLARITY NEGATIVE
    VPOLARITY POSITIVE
```





**VRTdsp** <arg>

Arguments: `LOG:<num>, LIN:<num>, FM:<num>, EXTernal:<num>`

This is a command with a single linked argument that specifies the vertical scale factor for the indicated display mode. The `DSRC` command must be used to enter `FM` or `EXTernal` modes, although the `VRTdsp` command still sets the scale factor.

| Link | <num> | Units | Display Type |
|---|---|---|---|
| `LOG:` | 10, 5, 1 | dB/div | logarithmic |
| `LIN:` | 8.839 to 279.5* | uV/div mV/div | linear |
| `FM:` | 10, 5, 1 | kHz/div | linear |
| `EXTernal:` | 17.5,87.5,175 | mV/div | linear |

* corresponds to reference levels of –70 dBm to +20 dBm

If units are not supplied, `LIN` assumes volts (for example, `LIN:.1 = LIN:100 Mv`), `FM` assumes Hertz, and `EXTernal` assumes volts.

```
VRTdsp LOG:5 (for example)
VRTdsp LIN:50 Uv (for example)
VRTdsp FM:5 kHz (for example)
VRTdsp EXTernal:175E-3 (for example)
```

**VRTdsp?** <arg>

Arguments: None, `LOG, LIN, FM, EXTernal`

This is a query with one or no argument that returns a linked response. The current scale factor is returned when `VRTdsp?` is used without an argument. When used with an argument, `VRTdsp?` returns the scale factor used when the indicated mode was last entered. Units are decibel (dB) for `LOG`, volts for `LIN` or `EXTernal`, and Hertz for `FM`.

```
VRTdsp?
    VRTDSP LOG:5 (for example)
VRTdsp? LOG
    VRTdsp? LOG:5 (for example)
```

```
VRTdsp? LIN
    VRTDSP LIN:50.0E-3 (for example)
VRTdsp? FM
    VRTDSP FM:5.E+3 (for example)
VRTdsp? EXTernal
    VRTDSP EXTERNAL:175.E-3 (for example)
```

**VSYnc** <arg> (Requires Option 10, Video Monitor)

Arguments: NEGative, POSitive

This single-argument command specifies the polarity of the video sync to be received with the Video Monitor mode.

```
VSYnc NEGative
VSYnc POSitive
```

**VSYnc?**

Arguments: None

This simple query returns the currently specified video sync polarity.

```
VSYnc?
    VSYNC NEGATIVE
    VSYNC POSITIVE
```

**WAIt**

Arguments: None

This is a command that requires no argument. It causes the spectrum analyzer to wait for an end -of-sweep to occur before processing any more commands. WAIt can be cancelled by the DCL or SDC GPIB commands, or the RS-232 BREAK command. See also the EOS command.

```
WAIt
```

**WAVfrm?**

Arguments: None

This simple query is functionally equivalent to the combination of the WFMpre? and CURve? queries. The WAVfrm header is never returned, even if HDR is ON.

```
WAVfrm?
```





**WFMpre** <arg>

Arguments: ENCdg:Asc, ENCdg:Bin, ENCdg:Hex, WFId:A, WFId:B, WFId:C, WFId:D

This is a command with one or more linked arguments. It is used to designate the source/destination register for the CURve command/query, and the data encoding to be used during a CURve transfer. Multiple arguments separated by commas (,) may be used. See the CURve command for an explanation of data encoding.

In its simplest form an argument(s) always follows the command header. These are examples of the general form to be used:

```
WFMpre WFID:<register>
WFMpre ENCdg:<type>
WFMpre WFID:<register>,ENCdg:<type>
```

where:

<register> = A, B, C or D

<type> = Asc, Bin, or Hex

For instance, these are all possible WFMpre commands:

```
WFMpre WFId:A
WFMpre WFId:B
WFMpre WFId:C
WFMpre WFId:D
WFMpre ENCdg:Asc
WFMpre ENCdg:Bin
WFMpre ENCdg:Hex
WFMpre WFId:D,ENCdg:Asc
```

The last command string is a typical message. In this example it indicates that register D is the source/destination for future CURve transfers, and that ASCII encoding is to be used for the data.

**WFMpre?** <arg>

Arguments: None, ENCdg, WFId

This is a query with one or no argument whose response provides information necessary for these CURve operations.

- Determine the currently selected source/destination register for CURve transfers
- Determine the data encoding for CURve transfers
- Interpret the result of a CURve query

When WFMpre? is used with the WFId argument, the query returns the currently selected source/destination register.

```
WFMpre? WFId

    WFMPRE WFID:B (for example)
```

When WFMpre? is used with the ENCdg argument, the query returns the currently selected CURve data encoding.

```
WFMpre? ENCdg

    WFMPRE ENCDG:ASC (for example)
```

When WFMpre? is is issued without an argument, it returns all the information necessary to interpret the response to a CURve? query. Following is an example of the response with HDR ON:

```
WFMPRE WFID:<register>,ENCDG:<type>,
NR.PT:512,PT.FMT:Y,PT.OFF:<nr1>,XINCR:<nr3>,
XZERO:<nr3>,XUNIT:<xunit>,YOFF:<nr1>,
YMULT:<nr3>,YZERO:<nr3>,YUNIT:<yunit>,
BN.FMT:RP,BYT/NR:1,BIT/NR:8,CRVCHK:
CHKSM0, BYTCHK:None
```

The response identifies the waveform, specifies data encoding, and provides the offsets, scale factors, and units necessary to plot or interpret the curve data. Tables 4-5 and 4-6 define the terms in the response.





**Table 4-5. Arguments of the WFMpre? Query.**

| Argument | Name | Description |
|---|---|---|
| 1 WFID:<id> | Waveform ID | ID may be A, B, C or D. |
| 2 ENCDG:<enc> | Encoding | The possibilities are ASC, BIN or HEX. |
| 3 NR.PT:512 | Number of points | There are 512 data points on every 2712 curve. |
| 4 PT.FMT:Y | Point format | Only Y-data is transmitted. X-data is implicit by the position of the point. |
| 5 PT.OFF:<nr1> | Point offset | See following formulas. |
| 6 XINCR:<nr3> | X increment | See following formulas. |
| 7 XZERO:<nr3> | X zero | See following formulas. |
| 9 XUNIT:<xunit> | X units | HZ (Hertz) or S(seconds). |
| 10 YOFF:<nr1> | Y offset | See following formulas. |
| 11 YMULT:<nr3> | Y multiplier | See following formulas. |
| 12 YZERO:<nr3> | Y zero | See following formulas. |
| 13 YUNIT:<yunit> | Y units | DBM, DBMV, DBV, DBUV, DBUW, DBUVM, V or HZ. |
| 14 BN.FMT:RP | Binary format | A binary positive integer. |
| 15 BYT/NR:1 | Bytes per number | Always 1 byte per number. |
| 16 BIT/NR:8 | Bits per number | Always 8 significant bits. |
| 17 CRVCHK:CHKSM0 | Curve checksum | Last byte of a binary or hex transfer is 2's complement of the modulo-256 sum of bytes following header that starts block; header is % (binary block); #H (ascii hex block). |
| 18 BYTCHK:NONE | Byte check | No per-byte error checking. |

**Table 4-6. Related Formulas.**

Let <valN> = value of the $N^{th}$ data point of the CURVE query. Then the X-value of that point is computed as:

$$XN = XZERO + XINCR * (N - PT.OFF).$$

The Y-value is computed as:

$$YN = YZERO + YMULT * (<valN> - YOFF)$$

Below is a WFMpre? query and the response obtained for the factory default power-up settings:

```
WFMpre?
    WFMPRE WFID:A,ENCDG:BIN,NR.PT:512,
    PT.FMT:Y,PT.OFF:5,XINCR:3.6e+6,XZERO:0.000,
    XUNIT:HZ,YOFF:245,YMULT:3.333E-1,YZERO:
    20.000E+0,YUNIT:DBM,BN.FMT:RP,BYT/NR:1,
    BIT/NR:8,CRVCHK:CHKSM0,BYTCHK:NONE;
```

*(for example)*

Using the preceding preamble (WFMPRE WFID:A,ENCDG.....), we will compute the value in engineering units of a data point within a CURve? response. In this example we have chosen the $255^{th}$ point and have assumed that the CURve? response indicates an integer value of 125 (curve data always have integer values). From the formulas above, the X and Y values of the point are given by these expressions:

$$XN\ (in\ xunits) = XZERO + XINCR * (N - PT.OFF)$$

and

$$YN\ (in\ yunits) = YZERO + YMULT * (VALN - YOFF)$$

where N = 255 and VALN = 125.

This data is extracted from the preamble:

| | |
|---|---|
| XZERO = 0 | YZERO = $2.0 \times 10^{1}$ |
| XINCR = $3.6 \times 10^{6}$ | YMULT = $3.333 \times 10^{-1}$ |
| PT.OFF = 5 | YOFF = 245 |
| XUNIT = HZ | YUNIT = DBM |





Evaluating the expressions, we find these results:

$$XN = 0 + 3.6 \times 10^{6} * (255 - 5) = 900 \times 10^{6}\ Hz$$

$$YN = 20 + .3333 * (125 - 245) = -20\ DBM$$

These results represent the center of the screen for the factory default power-up settings.

**ZERosp** <arg>

Arguments: OFF, ON

This single-argument command turns the zero span mode on and off.

```
ZERosp ON
ZERosp OFF
```

**ZERosp?**

Arguments: None

This simple query returns the current on/off status of the zero span mode.

```
ZERosp?
    ZEROSP ON
    ZEROSP OFF
```

# Section 5 — Status Reporting

Replace this page with the tab divider of the same name.

# SECTION 5
# STATUS REPORTING

Status reporting works much the same way for either interface (GPIB or RS-232) available on the 2711 and 2712 spectrum analyzers. The following discussion applies primarily to the GPIB; RS-232 protocol does not include device serial polling or an SRQ function.

If your instrument has the RS-232 port, we suggest that you review the entire section to understand the function of the status byte and its relationship to SRQ, RQS and serial polling. An additional subsection entitled ***RS-232 Error Reporting***, located at the end of this section, provides RS-232 status reporting information.

The 2711 and 2712 reports their status to the controller by means of the status byte and event codes. The status byte is a reporting feature provided in the IEEE-488 standard. When the spectrum analyzer is serially polled by the controller, it places a byte representing its general condition on the data bus. This status byte is then read by the controller. Event codes are generated by the spectrum analyzer for transmission to the controller over the data bus in response to an instrument-specific `EVEnt?` or `ERr?` query.

**NOTE**

*The 2711 and 2712 supports serial polling only; they do not support parallel polling.*

The status byte and event codes can indicate normal or abnormal conditions, although they typically alert the system user to abnormal conditions. Decoding of the status byte often provides the most efficient approach to detecting general classes of instrument conditions, but event codes provide more detailed information. For instance, suppose the signal on the spectrum analyzer screen is larger than the current reference level, and you attempt to use the `MMAx` command to place the marker on the signal peak. The status byte following the command (1110 0000) only indicates that a "device-dependent failure or warning" condition exists, while the event code (810) indicates "signal out of range".





Two techniques are available for monitoring the status of the spectrum analyzer at any time.

- Issue an `EVEnt?` or `ERr?` query and interpret the numerical response.
- Serially poll the instrument and decode the status byte.

When an instrument supporting the GPIB service request function (all Tektronix GPIB instruments do) detects an abnormal condition, it asserts (pulls to its low state) the dedicated service request (SRQ) line of the GPIB interface management bus, indicating that it requires attention. Therefore, to classify abnormal conditions, you can construct a subroutine which monitors the SRQ line. Either of the foregoing techniques can then be used to determine the nature of the event causing the SRQ.

## THE SERVICE REQUEST

The 2711 and 2712 spectrum analyzers have complete support for the IEEE-488 service request function (see ***Appendix A***). SRQs may be created by an instrument on the GPIB in response to certain equipment states including all abnormal conditions. However, SRQ generation may also be disabled with the `RQS` command. In addition to this general capability for inhibiting SRQs, certain SRQs may be independently masked.

The following events can generate SRQs.

- Press the front panel key sequence [UTIL] [6]. This is the "user request" event. The resulting SRQ cannot be independently masked.
- Completion of certain operations including End-Of-Sweep, normalization, User-Defined routine, signal search, plot, or ensemble average. Intermediate operation complete events are blocked. The plot complete event is generated after the plot is formatted and sent to the plotter; not necessarily when the plotter is finished. These SRQs are masked using the `EOS` command.
- Device-dependent failures or warnings generate SRQs. These SRQs include all spectrum analyzer error messages that may appear on-screen. They cannot be independently masked.

- Powering up the spectrum analyzer generates an SRQ when the power-up SRQ is enabled (requires GPIB interface). This mode is enabled or disabled by pressing the key sequence {UTIL} [4] [0] [0] [2].
- Illegal commands (command, execution, or internal errors) produce SRQs that cannot be independently masked.
- Failure of find-signal commands `MMAx`, `MRGTnx`, `MLFtnxt`, `HRAmpl`, `LRAmpl` generates an SRQ. This SRQ can be masked with the `SGErr` command.
- Crossing the display line limit generates an SRQ. This SRQ is enabled when the display line limit detector is on (`DLLimit`). This SRQ cannot be independently masked.
- Internal hardware/firmware errors generate an SRQ that cannot be independently masked.

## STATUS BYTE

The status byte usually provides information about instrument conditions according to certain categories (normal, abnormal, busy, command error, execution error, etc.). This is different from data returned by the `ERr?` and `EVEnt?` queries, which provide detailed information about the cause of the current status. For instance, the response to an `EVEnt?` query might describe what kind of error or warning prompted the spectrum analyzer to assert `SRQ` and report abnormal status.

The status byte is stored in the status byte register, which is updated as conditions in the spectrum analyzer change. The spectrum analyzer responds to a serial poll by placing the status byte on the data bus (see ***Appendix A***) to be read by the controller. Only the most recent status byte is placed on the bus. The status byte is cleared by a serial poll of the spectrum analyzer, by the DCL GPIB command, or (if the instrument is first addressed) the SDC GPIB command.

Status bytes can be divided into two categories: device-dependent and system. Device-dependent status bytes represent conditions unique to the spectrum analyzer. System status bytes have a fixed definition for all Tektronix devices and identify most events common to any IEEE-488 system. Bit 8 is always set (1) for device-dependent status bytes and is always reset (0) for system status bytes. Tables 5-1 through 5-4 show general and specific encodings of device-dependent and system status bytes.





**Table 5-1. General System Status Bytes.**

| Status Byte Bits | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| Bit Value | 0 | R | E | B | S | S | S | S |

Where:

R = SRQ pending (1= pending, 0 = not pending)

B = Instrument busy (1= busy, 0= not busy)

S = System status

E = Normal (0)/Abnormal (1)

**Table 5-2. General Device-dependent Status Bytes.**

| Status Byte Bits | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| Bit Value | 1 | R | D | D | D | D | D | D |

Where:

R = SRQ pending (1= pending, 0 = not pending)

D = Instrument-specific

Because of the status coding scheme, only one condition can be reported in a single status byte. More than one condition may exist in the spectrum analyzer at any given time. Therefore, the following rules are used to determine which condition is reported by any status byte.

- Each condition is assigned a priority according to Table 5-5. Only one occurrence of each condition is retained in memory.
- When `RQS` is `ON` the status byte following an SRQ represents the condition that caused the SRQ (not necessarily the highest priority).
- When `RQS` is `OFF` the normal device-dependent status shown in Table 5-3 is reported.
- After a status byte is read by the controller, the status byte is updated with the highest priority condition which has not yet been reported.
- All status conditions that have occurred but have not been reported (except power on) can be cleared by a device clear (DCL) command.
- Bit 5 is the current status of the message processor.

**Table 5-3. Specific System Status Bytes.**

| Status Byte Bits | | | | | | | | Byte* Value | Event Description |
|---|---|---|---|---|---|---|---|---|---|
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | |
| 0 | 0 | 0 | B | 0 | 0 | 0 | 0 | 00,10 | No status to report |
| 0 | 1 | 0 | B | 0 | 0 | 0 | 1 | 41,51 | Power on |
| 0 | 1 | 0 | B | 0 | 0 | 1 | 1 | 43,53 | User Request |
| 0 | 1 | 1 | B | 0 | 0 | 0 | 1 | 61,71 | Command error |
| 0 | 1 | 1 | B | 0 | 0 | 1 | 0 | 62,72 | Execution error |
| 0 | 1 | 1 | B | 0 | 0 | 1 | 1 | 63,73 | Internal error |

* Byte value depends on B = 1 or B = 0

**Table 5-4. Specific Device-dependent Status Bytes.**

| Status Byte Bits | | | | | | | | Byte* Value | Event Description |
|---|---|---|---|---|---|---|---|---|---|
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | |
| 1 | 0 | U | B | N | S | A | P | | Normal device-dependent status |
| 1 | 1 | 0 | B | 0 | 0 | 1 | 0 | C2,D2 | Device-dependent operation complete (EOS, Norm, etc.) |
| 1 | 1 | 1 | B | 0 | 0 | 0 | 0 | E0,F0 | Device-dependent failure/warning |
| 1 | 1 | 1 | B | 0 | 0 | 1 | 1 | E3,F3 | Signal find error (MFBIG, MRGTNX, etc.) |
| 1 | 1 | 1 | B | 0 | 1 | 0 | 0 | E4,F4 | Display line limit exceeded |
| 1 | 1 | 1 | B | 0 | 1 | 0 | 1 | E5,F5 | Firmware error |

* Byte value depends on B = 1 or B = 0

Where:

U = UDP executing (1)
N = Normalization(s) executing (1)
S = Signal search executing (1)
A = Ensemble average executing (1)
P = Plot executing (1)
B = Busy (1)





**Table 5-5. Event Priorities.**

| Event | Priority* |
|---|---|
| Power on | 1 |
| Command error | 2 |
| Execution error | 3 |
| Internal error | 4 |
| User request | 5 |
| Signal find error (MFBIG, MRGTNX, etc.) | 6 |
| Display line limit exceeded | 6 |
| Device-dependent failure or warning | 7 |
| Device-dependent operation complete (EOS, Norm, etc.) | 8 |
| Firmware error | 9 |
| Normal device dependent status | 10 |
| No status to report | 10 |

* Highest priority = 1

Example 5-1 shows a QuickBASIC subroutine that can be used with the National Instruments GPIB board and software to report the current status byte. The first five lines must be part of the parent program. `IBFIND` and `IBRSP` are callable subroutines supplied by National Instruments. See Section 6, ***Programming***, for explanations and additional programming instructions.

**Example 5-1. Subroutine to Read the Status Byte.**

```
REM $INCLUDE 'IBDCL4.BAS'
COMMON SHARED BDNAME$,BD%,SPR$
BDNAME$ = "TEK_SA"
CALL IBFIND (BDNAME$, BD%)
GOSUB SERIAL.POLL
        :
        :
SERIAL.POLL:
  CALL IBRSP (BD%,SPR%)
  PRINT SPR%
RETURN
```

## EVENT CODES

Not all GPIB applications need the capability provided by the SRQ function and the serial poll sequence. In fact, the SRQ service routine is often more complex than the application demands. For this reason the Request for Service (RQS) command is implemented in the 2711 and 2712. This command allows the controller to prevent the instrument from asserting SRQ. In this mode of operation, the EVEnt? and ERr? queries provide for the transmission of error and status information. The EVEnt? and ERr? commands return the same codes; ERr? is included for compatibility with Tektronix 490-Series spectrum analyzers.

The EVEnt? and ERr? queries provide more information about the cause of an event than the status byte does. For this reason the event query can be useful in the RQS ON and RQS OFF modes.

Event codes are grouped into categories as shown in Table 5-6. Individual event codes are listed in Table 5-7.

Event codes are assigned priorities according to Table 5-5, but only the first event code of a given priority is accumulated in the pending event table. However, the EVEnt? and ERr? queries return a single code. Therefore, to ensure that all pending event codes are reported, you must continue to issue EVent? or ERror? queries until event code zero (0) is returned.

**Table 5-6. Event Code Categories.**

| Numeric Range | Event |
|---|---|
| 0 - 99 | Local events (not used) |
| 100 - 199 | Command errors |
| 200 - 299 | Execution errors |
| 300 - 399 | Internal errors |
| 400 - 499 | System events |
| 500 - 599 | Execution warnings (not used) |
| 600 - 699 | Internal warnings (not used) |
| 700 - 899 | Spectrum analyzer dependant events |





When RQS is OFF.EVent? or ERror? returns the highest priority event in the table. After RQS has been turned ON, either query returns the code corresponding to the event reported in the status byte (not necessarily the highest priority).

When an event code is read the code is also cleared from the pending event table, but this does not clear the status byte. In a similar manner, reading a status byte clears it from the table, but the event code is not cleared. In either RQS mode, the DCL or (if the instrument is first addressed) SDC GPIB commands may also be used to clear all event codes except POWER ON.

Example 5-2 shows a QuickBASIC subroutine that can be used with the National Instruments GPIB board and software to report all pending event codes. However, when RQS is ON, you must first call SERIAL.POLL to ensure a valid event code is returned. Further, the response header must be turned off so EVENT.CODE$ is returned exclusively as a number string. IBFIND, IBWRT, and IBRD are callable subroutines supplied by National Instruments. See Section 6, ***Programming***, for explanations and additional programming instructions.

**Example 5-2. Subroutine for Reading Event Codes.**

```
REM $INLUDE 'IBDCL4.BAS'
COMMON SHARED BDNAME$, BD%,EVENT.CODE$
BDNAME$ = "TEK_SA"
CALL  IBFIND (BDNAME$, BD%)
GOSUB EVENT.FIND
      :
EVENT.FIND:
  WRT$ = "HDR OFF;EVE?"
  DO
     CALL IBWRT (BD%,WRT$)
     CALL IBRD (BD%,EVENT.CODE$)
     PRINT EVENT.CODE$
  WHILE VAL (EVENT.CODE$) <> 0
RETURN
```

## RS-232 ERROR REPORTING

The RS-232 protocol does not contain a mechanism that duplicates the GPIB SRQ function as described earlier in this section. To fill this need, the RS-232 configuration supplies a settable VERBOSE mode as an alternative. When VERBOSE mode is on, every command is guaranteed a response. Three response types are possible:

- The string "OK"; returned for a successful command
- A query response; returned for a successful query
- ERR *n*; returned when error number *n* is detected

Errors reported while VERBOSE mode is on have no effect on the status reporting structure described earlier for GPIB. The RS-232-specific query `STByte?` will return the GPIB serial poll response byte for analysis. Because `STByte?` is a normal query, the Busy bit (bit 5 of the status byte) is always reported to be ON.

A query is the only means for returning information to the interface when VERBOSE mode is off. In this mode, the user must explicitly issue an `EVENT?` or `STBYTE?` query to retrieve error information.

The REQUEST indicator on the spectrum analyzer's display screen indicates when an error is pending. If `RQS` is `ON` and an error is pending, the REQUEST indicator appears on-screen. The programmer must send a `STByte?` query (simulating the return of the GPIB serial poll response) followed by either `EVEnt?` or `ERR?` to report and clear the error condition. If `RQS` is `OFF` and an error is pending, the REQUEST indicator does not appear on the spectrum analyzer screen. Under these conditions an `EVEnt?` or `ERR?` query is required to report and clear the error condition. Otherwise the error remains pending.

To use the VERBOSE feature a routine must be set up that reads each possible response and sends each response type for parsing and possible processing. See the RS-232 sample program in Section 6, ***Programming***. This routine uses VERBOSE mode in place of GPIB SRQ.





Three types of errors can occur that are related to problems with RS-232 communications: parity, framing and overrun. Parity and framing errors may occur because of mismatch between configuration settings (baud rate, parity, etc.) or because of noise. Overrun is more likely due to a design problem. For example, the inability to handle interrupts at the required rate results in overrun.

When an error occurs, the appropriate event is declared and all unparsed data are discarded until a terminator is received. Table 5-7 lists the complete set of 2711 and 2712 event codes and status bytes, including three RS-232-specific event codes: numbers 410, 411 and 412.

The status bytes listed in Table 5-7 assumes that the Busy bit is off. When the Busy bit is on, add values of 16 (decimal) or 10 (hexadecimal) to the table entry.

Table 5-7. Event Codes.

| Event Code | Status Byte Dec | Status Byte Hex | Event Description |
|---|---|---|---|
| 0 | 128 | 80 | No device-dependent status to report |
| 0 | 0 | 00 | No system status to report |
| 101 | 97 | 61 | Command header error |
| 102 | 97 | 61 | Header delimiter error |
| 103 | 97 | 61 | Command argument error |
| 104 | 97 | 61 | Argument delimiter error |
| 105 | 97 | 61 | Non-numeric Arg.(numeric expected) |
| 106 | 97 | 61 | Missing argument |
| 107 | 97 | 61 | Invalid message unit delimiter |
| 108 | 97 | 61 | Binary block checksum error |
| 109 | 97 | 61 | Binary block byte count error |
| 121 | 97 | 61 | Illegal Hex Character |
| 122 | 97 | 61 | Unrecognized argument type |
| 123 | 97 | 61 | The argument is too large |
| 124 | 97 | 61 | Non-binary Arg. (binary or hex expected) |
| 151 | 97 | 61 | Illegal response value in query |
| 201 | 98 | 62 | Remote command received when in local mode |
| 202 | 98 | 62 | Command aborted - (rtl) return to local |
| 203 | 98 | 62 | I/O Deadlock detected |
| 205 | 98 | 62 | Argument out of range |
| 206 | 98 | 62 | Group execute trigger ignored |
| 252 | 98 | 62 | System error (Illegal command) |
| 253 | 98 | 62 | Integer overflow (range 0-65535) |
| 371 | 99 | 63 | Output buffer full (too many queries) |
| 372 | 99 | 63 | Input buffer full (command too Long) |
| 401 | 65 | 41 | Power On |
| 403 | 67 | 43 | User Request |
| 410 | 99 | 63 | RS-232 Parity error |
| 411 | 99 | 63 | RS-232 Framing error |
| 412 | 99 | 63 | RS-232 Hardware overrun |
| 700 | 224 | E0 | Error |





Table 5-7. (Continued)

| Event Code | Status Byte Dec | Status Byte Hex | Event Description |
|---|---|---|---|
| 701 | 224 | E0 | Illegal Parameter Passed |
| 704 | 224 | E0 | Illegal Command |
| 705 | 224 | E0 | malloc: Ran out of memory |
| 706 | 224 | E0 | RunTask: Cannot Start process |
| 707 | 224 | E0 | Interrupt Fault at FF |
| 708 | 224 | E0 | Interrupt Fault |
| 709 | 224 | E0 | Command Not Implemented |
| 710 | 224 | E0 | Markers are Off |
| 711 | 224 | E0 | Signal Cannot Be Set Properly |
| 712 | 224 | E0 | No Signal at Counter Input |
| 713 | 224 | E0 | Counter frequency unstable |
| 714 | 224 | E0 | Normalization Suggested |
| 715 | 224 | E0 | Timer Interrupt Fault |
| 716 | 224 | E0 | No Signal (Normalizations) |
| 717 | 224 | E0 | Ampl out of range (Normalizations) |
| 718 | 224 | E0 | Freq out of range (Normalizations) |
| 719 | 224 | E0 | Func Not Avail in Current Mode |
| 720 | 224 | E0 | Frequency Normalization Failed |
| 721 | 224 | E0 | Amplitude Normalization Failed |
| 722 | 224 | E0 | Reference Normalization Failed |
| 723 | 224 | E0 | Internal Ref Freq too inaccurate |
| 724 | 224 | E0 | Internal Ref Ampl too inaccurate |
| 725 | 224 | E0 | Selected Stored Setting is Empty |
| 726 | 224 | E0 | Video Monitor Not Installed |
| 727 | 224 | E0 | Satellite Video Monitor Not Installed |
| 728 | 224 | E0 | Not Installed |
| 729 | 224 | E0 | Counter Not Installed |
| 730 | 224 | E0 | Cannot overwrite saved display |
| 731 | 224 | E0 | NVM Checksum Error |
| 732 | 224 | E0 | Non-Compatible NVM Format |
| 733 | 224 | E0 | First Step Must Be Done First |
| 734 | 224 | E0 | FREQ Norm Suggested (Inner PLL) |

Table 5-7. (Continued)

| Event Code | Status Byte Dec | Status Byte Hex | Event Description |
|---|---|---|---|
| 735 | 224 | E0 | FREQ Norm Suggested (Set VCO) |
| 736 | 224 | E0 | Polynomial Has No Solution |
| 737 | 224 | E0 | Last Pwr Down Reg Checksum Err |
| 738 | 224 | E0 | Storage Register Empty |
| 739 | 224 | E0 | Normalized Result Out of Range |
| 740 | 224 | E0 | Function not avail. in LIN mode |
| 741 | 224 | E0 | Cannot Store - NV Memory Full |
| 742 | 224 | E0 | AMPL Norm Suggested (VR Pin DAC) |
| 743 | 224 | E0 | Cannot Calc. Vert. Sensitivity |
| 744 | 224 | E0 | Cannot Count (VCO,IF) |
| 745 | 224 | E0 | Cannot Normalize PLL VCO |
| 746 | 224 | E0 | Cannot Count Beat Frequency |
| 747 | 224 | E0 | FREQ Norm Suggested (Set Beat) |
| 748 | 224 | E0 | FREQ Norm Suggested (1st LO) |
| 749 | 224 | E0 | Setting Corrupted |
| 750 | 224 | E0 | NVM Fragmentation Err |
| 751 | 224 | E0 | NVM Segmentation Error |
| 752 | 224 | E0 | Comm Port Not Installed |
| 753 | 224 | E0 | Real Time Clock Hardware Failure |
| 754 | 224 | E0 | Real Time Clock Not Installed |
| 755 | 224 | E0 | FREQ Norm Suggested (Find Side) |
| 756 | 224 | E0 | FREQ Norm Suggested (Span DAC) |
| 759 | 224 | E0 | Insufficient Memory Available |
| 760 | 224 | E0 | Not Avail in Short Holdoff Mode |
| 761 | 224 | E0 | Short Holdoff Mode Not Installed |
| 762 | 224 | E0 | Cannot Overwrite Stored Setting |
| 763 | 224 | E0 | Cannot Overwrite Stored Waveform |
| 764 | 224 | E0 | Delete Existing Program First |
| 765 | 224 | E0 | Editing Buffer Is Empty |
| 766 | 224 | E0 | Remove Protection First |
| 767 | 128 | 80 | Wait Aborted, Sweep Not Armed |
| 768 | 224 | E0 | Selected Program Is Empty |





Table 5-7. (Continued)

| Event Code | Status Byte Dec | Status Byte Hex | Event Description |
|---|---|---|---|
| 769 | 224 | E0 | Program Not Executable |
| 770 | 224 | E0 | Not Avail in Waterfall Mode |
| 771 | 224 | E0 | Amplitude out of Calibration |
| 772 | 224 | E0 | Illegal Start/Stop/Inc Values |
| 773 | 224 | E0 | Delete Existing Table First |
| 774 | 224 | E0 | Selected Table Is Empty |
| 775 | 224 | E0 | Use ANTENNA SETUP Menu First |
| 776 | 224 | E0 | Table is too large to Edit |
| 777 | 224 | E0 | Default Data Loaded |
| 778 | 224 | E0 | Delete Editing Buffer First |
| 779 | 224 | E0 | Warning: Using Empty Ant Table |
| 780 | 224 | E0 | Not Available with DBUV/M Idle |
| 781 | 224 | E0 | Mkr Would Overwrite Noise Value |
| 782 | 224 | E0 | Function Not Avail in DBUV/M Mode |
| 783 | 224 | E0 | No Listener |
| 784 | 224 | E0 | Select TALK ONLY mode first |
| 785 | 224 | E0 | Tracking Generator Norm. Failed |
| 786 | 224 | E0 | QP Filters Not Installed |
| 787 | 224 | E0 | Destination waveform conflict |
| 788 | 224 | E0 | TG normalization suggested |
| 789 | 224 | E0 | EMC mode must be active |
| 790 | 229 | E5 | Input buffer empty (Firmware error) |
| 791 | 229 | E5 | Illegal event code (firmware error) |
| 792 | 229 | E5 | Illegal command received from CP |
| 793 | 229 | E5 | Illegal byte count in command |
| 800 | 224 | E0 | Exiting Quasi-Peak Detector |
| 801 | 224 | E0 | Out Of Range |
| 802 | 224 | E0 | None of the Traces are Active |
| 803 | 224 | E0 | Uncal Off |
| 804 | 224 | E0 | Uncal On |
| 805 | 128 | 80 | Single Sweep Mode |

Table 5-7. (Continued)

| Event Code | Status Byte Dec | Status Byte Hex | Event Description |
|---|---|---|---|
| 806 | 128 | 80 | Single sweep armed |
| 807 | 128 | 80 | Single sweep trigger |
| 808 | 224 | E0 | No signal Found Above Threshold |
| 809 | 224 | E0 | Inactive marker off screen |
| 810 | 224 | E0 | Signal Over Range |
| 811 | 224 | E0 | Function Not Avail in Max Span |
| 812 | 224 | E0 | Ref level at new range limit |
| 813 | 224 | E0 | Normalization Complete |
| 814 | 224 | E0 | No signal at center of display |
| 815 | 224 | E0 | Not Avail w/ Display Storage On |
| 816 | 224 | E0 | 500 kHz RBW used for Counting |
| 817 | 224 | E0 | Noise Level Less Than 2 dB |
| 818 | 224 | E0 | Start Frequency Changed |
| 819 | 224 | E0 | Stop Frequency Changed |
| 820 | 224 | E0 | Signal out of IF passband |
| 821 | 224 | E0 | No Modulation on signal |
| 822 | 224 | E0 | 1st Measurement Complete |
| 823 | 224 | E0 | Disconnect Input Signal |
| 824 | 224 | E0 | ZERO SPAN Entered |
| 825 | 224 | E0 | Must Be In Delta Marker Mode |
| 826 | 224 | 80 | Stand By |
| 827 | 224 | E0 | Printer Error |
| 828 | 224 | E0 | Printer Out Of Paper |
| 829 | 224 | E0 | Printer Is Not Connected |
| 830 | 224 | E0 | Port Off Line |
| 831 | 128 | 80 | Formatting Plot |
| 832 | 224 | E0 | Plot Aborted |
| 833 | 224 | E0 | Can't Count With Corrections Off |
| 834 | 224 | E0 | Counter Signal Out of IF Passband |
| 835 | 224 | E0 | Vert Mode/Scale Mismatch on Diff |
| 836 | 224 | E0 | Query Not Available |
| 837 | 224 | E0 | Average Noise Too Low |





Table 5-7. (Continued)

| Event Code | Status Byte Dec | Status Byte Hex | Event Description |
|---|---|---|---|
| 838 | 224 | E0 | Only Waveforms Saved |
| 839 | 224 | E0 | Only Waveforms Deleted |
| 840 | 224 | E0 | File System Full |
| 841 | 224 | E0 | File System Directory Full |
| 842 | 224 | E0 | File Size Error |
| 843 | 224 | E0 | Too Many Files Open |
| 844 | 224 | E0 | File Not Found |
| 845 | 224 | E0 | Protected File |
| 846 | 224 | E0 | Cannot Delete File While in Use |
| 847 | 224 | E0 | Additional NVRAM Not Installed |
| 848 | 224 | E0 | Invalid File Number |
| 849 | 224 | E0 | Invalid Device Number |
| 850 | 224 | E0 | End of File |
| 851 | 224 | E0 | NVM Version Mis-Match |
| 852 | 224 | E0 | Fatal Error in File |
| 853 | 224 | E0 | Directory Error In File |
| 854 | 224 | E0 | Data Error in File |
| 856 | 128 | 80 | Clear Event |
| 857 | 224 | E0 | Calibrator Doesn't Match Readout |
| 858 | 128 | 80 | Return to Local Request |
| 859 | 224 | E0 | Display Line Off Screen |
| 860 | 224 | E0 | DBUV/M Measurement Mode Idle |
| 861 | 224 | E0 | Search Terminated, Max Signals |
| 862 | 128 | 80 | Lock Event |
| 863 | 128 | 80 | Unlock Event |
| 864 | 128 | 80 | DCL End |
| 865 | 128 | 80 | User Defined Program in Process |
| 866 | 128 | 80 | Plot in Process |
| 867 | 128 | 80 | Average in Process |
| 868 | 128 | 80 | Signal Search In Process |
| 869 | 128 | 80 | Normalizing |
| 880 | 194 | C2 | User Defined Program Complete |

Table 5-7. (Continued)

| Event Code | Status Byte Dec | Status Byte Hex | Event Description |
|---|---|---|---|
| 881 | 194 | C2 | Plot Complete |
| 882 | 194 | C2 | Ensemble Average Complete |
| 883 | 194 | C2 | Signal Search Complete |
| 884 | 194 | C2 | Normalization Process Finished |
| 885 | 194 | C2 | End of Sweep Detected |
| 895 | 228 | E4 | Display Line Limit Exceeded |
| 896 | 227 | E3 | Signal Find Error |

# Section 6 — Programming

**Replace this page with the tab divider of the same name.**

# SECTION 6
# PROGRAMMING

**NOTE**

*Program examples in this section are written for the 2712 spectrum analyzer. To modify these programs for the 2711 spectrum analyzer, simply replace* `2712` *with* `2711`.

This section discusses general approaches to GPIB and RS-232 programming on IBM-compatible MS-DOS/PC-DOS computers and their function-alike counterparts. The intent is not to teach you programming, but to provide a few useful subroutines and demonstration programs. Experienced programmers may not need to read this section.

There are no uniform standards for all GPIB boards and all programming languages. The use of RS-232 is more standardized across the IBM/PC family, however, the examples in this section will be both device and language specific.

You can benefit by reading the sections on GPIB programming even if you are programming for the RS-232 interface. We recommend that you first read GPIB-related information, then proceed to the RS-232 programming section for more specific information. The programming language used for either interface is Microsoft's QuickBASIC, version 4.5.

## INTRODUCTION TO GPIB PROGRAMMING

The routines in the ***GPIB Sample Routines*** section can also work with earlier versions of QuickBASIC. However, you must use the appropriate GPIB system software and make the changes to function calls as outlined in the `READ-QB.DOC` document file from National Instruments. The GPIB system software is that supplied by National Instruments along with their boards.

The devices considered are the National Instruments PCII, PCIIA, and PCII/IIA GPIB boards, supplied by Tektronix or directly available from National Instruments dealers. Consult Section 1, ***Introduction***, of this manual and the material supplied with your board for detailed instructions.





## BEGINNING YOUR GPIB PROGRAM

Before you begin programming, several steps must be performed to ensure QuickBASIC has the necessary GPIB information. See ***Preparing the Software*** in the ***Setting Up For GPIB Operation*** part of Section 1 for the necessary procedure.

Generally speaking, a BASIC or QuickBASIC program begins with a list of declarations. This establishes variable types and the names of global variables. A number of variable names are used by the National Instruments software. To prevent any confusion or misunderstanding regarding these variables, National Instruments supplies a program file which declares them for you. To properly declare the reserved National Instruments variables, place this line at the beginning of your programs:

```
REM $INCLUDE 'QBDECL4.BAS'
```

QuickBASIC supports the sub-program module concept, so traditional `COMMON` statements are replaced with `COMMON SHARED` statements. This means that the variable names so declared may be shared by all modules. Further, each sub-program module must be declared. The following lines declare the global variables used by our subroutines and demonstration program:

```
COMMON SHARED BD%,BDNAME$,RD$,
WRT$ , EVENT.CODE$
COMMON SHARED NUMBYT%
```

Table 6-1 lists the global variables declared by `QBDCL4.BAS` and the `COMMON SHARED` statements, and their purpose.

You may desire to dimension arrays or variables at the beginning of your program, even though they are to be used later. For instance, `CUR%()` is the integer array used for curve data in our demonstration program. `RD$` is a string variable that we use as the destination for a variety of data transfers.

Table 6-1. Variable Names.

| Name | Description | Remarks |
|---|---|---|
| BD% | Integer value associated with a particular device name by IBFIND(BDNAME$,BD%). | Other names can be used such as MYDEV%, FRST.DEV%, A1%, etc. |
| BDNAME$ | String variable set to a device name such as TEK_SA. The name is the one used when you set up the device with IBCONF. | Any string variable name can be used such as MYNAME$, DEVNAM$, DN$, etc. |
| EVENT.CODE$ | A string variable argument of IBRD containing the 2711 or 2712 event code. | Any string variable name can be used: (MYNAME$, DEVNAM$, DN$, etc.) |
| IBCNT% | Integer variable updated by GPIB system software after a read, write or command: the number of bytes transferred. | This name is reserved. |
| IBERR% | Integer variable returned by GPIB system software when error bit of status word is set. Range: 0 to 7, or 10 to 16. | See your GPIB manual for meanings of IBERR% values. The name is reserved. |
| IBSTA% | Integer variable updated by all GPIB system software functions. | Refer to ***Status Byte*** in Section 5, ***Status Reporting***. |
| NUMBYT% | Integer argument of DEBLK which indicates the number of bytes converted. | Any integer variable name can be used (MYNUM%, INTEG%, etc.). |
| RD$ | String variable used with the IBDR function to contain the data returned by the 2711 or 2712. | Other names can be used such as RET.DAT$, MYDAT$, S1$, etc. |
| WRT$ | A string variable used with the IBWRT function to contain the data to be sent to the 2711 or 2712. | Other names can be used such as WRT.DAT$, MESSAGE$, B4$, etc. |





If memory size is not a limiting factor in your controller, you may wish to set up the maximum size of RD$ at the head of the program with a SPACE$ command as in this example.

```
DIM SHARED CUR%(512)
RD$ = SPACE$(<max anticipated size>)
```

The number of bytes in the response is less than 5000 for all queries except PLOT? and FILE?.

The PLOT? query may require as much as 37,000 bytes for HPGL plots and 61,100 for Epson plots. The number of points in a PLOT? response depends on the number of waveforms present and the acquisition mode.

A user-defined program (UDP) can theoretically occupy all available memory. As a result, UDPxx files larger than 100,000 bytes are possible. However, 5000 bytes are usually adequate for all but the largest UDPs.

## ERROR TRAPPING

There are three types of errors that may can occur in your program: DOS, GPIB System, and instrument-related. Different techniques are used to deal with each type. Errors are typically trapped so corrective action can be taken, or the program is caused to end in a non-destructive manner (gracefully).

### DOS Errors

DOS errors may occur on instruments equipped with either the GPIB or RS-232 interface. DOS errors typically happen when trying to access a device that is missing or not ready. They are traditionally handled with the BASIC ON ERROR statement. Following the declarations at the head of the program, you must add this line:

```
ON ERROR GOTO ERR.TRAP
```

Next you should construct a subroutine to deal with the problem. This routine may be used to end the program in a non-destructive manner (gracefully):

```
ERR.TRAP:
    PRINT "DOS ERROR HAS OCCURRED."
    PRINT "CHECK YOUR SYSTEM SETUP."
    END
RESUME NEXT
```

With this routine the program stops if a DOS error is encountered. You simply check your system and restart. Consult a BASIC programming manual for selective error trapping techniques and other approaches.

### GPIB System Errors

Instruments with a GPIB interface may encounter GPIB system errors. (See ***Sample RS-232 Controller*** later in this section describes error handling for RS-232.) These errors are detected by examining IBSTA% following each GPIB function call.

A GPIB error has occurred whenever the value of IBSTA% is equal to or greater than 32768. To determine what type of error has occurred, you can check the value of IBERR%. Once you have verified the type of error you may take corrective action, proceed without doing anything (if the program continues to run and yields valid results), or end the program gracefully. This subroutine can be called following each call to a GPIB function to determine the type of error and to end the program gracefully:

```
GPIB.ERR
    IF IBSTA%<32768 THEN RETURN
    PRINT "GPIB ERROR HAS OCCURRED"
    PRINT "GPIB ERROR CODE IS ";IBERR%
    END
RETURN
```

### Instrument-related Errors

A third type of error involves those directly related to the spectrum analyzer or its interface with the bus. Any time the spectrum analyzer detects an abnormal condition, it issues a service request (SRQ) if RQS is set to ON. The SRQ causes the GPIB board to generate a light pen interrupt.

You can automatically detect and decode abnormal events by branching to an interrupt handler whenever the interrupt occurs. To do this add these lines to your program:

```
ON PEN GOSUB ABNORM.EVE
PEN ON
        :
ABNORM.EVE:
    CALL SERIAL.POLL
    CALL EVENT.FIND
RETURN
```





**Table 6-2. GPIB System Software Callable Subroutines.**

| Subroutine | Description |
|---|---|
| DEBLK(CUR%(),CUR%(), CNT%,8,NUMBYT%) | Convert first CNT% elements of array CUR%() from binary block to 2-byte integer format in same array and return the number of bytes converted. |
| IBFIND(BD%,BDNAME$) | Open device indicated by BDNAME$ and return unit descriptor BD%. |
| IBRD(BD%,RD$) | Read data from BD% to string RD$. |
| IBRDF(BD%,FILENAME$) | Read data from device BD% and store to disk in file named <FILENAME$>. |
| IBRDI(BD%,CUR%(),CNT%) | Read CNT% data bytes from device BD% into integer array CUR%(). |
| IBRSP(BD%,SPR%) | Perform serial poll of device BD%. |
| IBSRE(BD%,V%) | Enable/disable remote mode in the device indicated by BD%. V%=0 disables. |
| IBWRT(BD%,WRT$) | Send contents of string variable WRT$ to device indicated by BD%. |
| IBWRTF(BD%,FILENAME$) | Send disk file named <FILENAME$> to device indicated by BD%. |

SERIAL.POLL and EVENT.FIND are sample subroutines discussed in Section 5, ***Status Reporting***.

## GPIB SYSTEM SOFTWARE

All of the programming examples in this manual make use of the subroutines supplied by National Instruments as part of the GPIB or Tektronix GURU II software. If you are using a board from another manufacturer, equivalent software should have accompanied your board. This software supplies the interface between the programming language and the installed GPIB board.

Table 6-2 lists the National Instruments system subroutines used in this manual. Consult your GPIB documentation for detailed information about these and many other subroutines that are commonly available.

# GPIB SAMPLE SUBROUTINES

This subsection contains a collection of subroutines that illustrate simple approaches to transferring various types of data between the system controller and the spectrum analyzer. You may wish to incorporate them into your own software or modify them so they are more appropriate to your needs.

The header statements in Example 6-1 can be placed at the beginning of a program, and be used as a basis for the subroutines in this section. If you modify the subroutines, or add new ones, the statements may no longer be adequate.

**NOTE**

*The header statements in Example 6-1 and the sample subroutines in this section do not make provisions for error trapping and event reporting.*

See Section 5, ***Status Reporting***, and the demonstration program at the end of this section for error and event reporting routines.

**Example 6-1. GPIB Program Header Statements.**

```
REM $INCLUDE: 'QBDECL4.BAS'
COMMON SHARED BD%,BDNAME$,RD$,WRT$
RD$=SPACE$(5000)
BDNAME$ = "TEK_SA"
CALL IBFIND (BDNAME$, BD%)
```

## Curve Transfers

Curves (waveforms) transferred from the spectrum analyzer to the controller are important for data analysis, archiving, and reporting purposes. Curves transferred to the spectrum analyzer can be used for comparison or to establish references. The CURve command transfers a block of data from the controller to the spectrum analyzer and the CURve? query returns a block of data from the spectrum analyzer to the controller.

The data block represents the 512 points in a 2711and 2712 waveform (see Section 4, ***Command and Query Definitions***, for CURve? response formats). Before curve data is transferred you must specify which digital display register (A, B, C, or D) the curve is coming from or going to, and the type of data encoding to be used. The encoding (ASCII-encoded decimal, ASCII-encoded hexadecimal, or binary) is determined by the waveform preamble (see the WFMpre command).





The destination of the transmitted data or origin of the returned data is also determined by the preamble unless the A, B, C, or D argument is used with the CURve? query. In these cases, the origin is specified by the argument. For example, CURve? C returns data from the C register.

Example 6-2 shows two QuickBASIC subroutines that can be used with the National Instruments board and software to send and return ASCII-encoded curve data. The returned data are displayed on the controller screen, and will resemble the ASCII example in the CURve command as described in Section 4. The WFMpre command determines the encoding and source or destination registers in both subroutines. An alternate approach to transferring curve data (as packed integers) is illustrated in the GPIB demonstration program at the end of this section.

The curves transferred to the spectrum analyzer in these examples are the previously returned waveforms, but you can also send artificially generated curves. Such curves can be generated in ASCII format using a spreadsheet. Curves should always be transferred to a saved register to ensure they are not immediately overwritten by the next spectrum analyzer sweep.

Whenever a curve is generated, the WRT$ = MID...line in the PUT.CURVE subroutine must be replaced by a statement such as this one:

```
WRT$ = "CURve "+IND$+BC$+DATA$+CK$
```

where:

IND$ = Null for ASCII, #H for hex, % for binary

BC$ = Null for ASCII, 0201 for hex, CHR$(2)+CHR$(1) for binary (Byte Count)

DATA$ = 512 data points, appropriately encoded, representing the curve

CK$ = Null for ASCII , HEX$(chksum) for hex, "0"+HEX$(chksum) if chksum < 16, CHR$(chksum) for binary (Checksum)

**Example 6-2. Subroutines to Return or Transmit Curve Data.**

```
    GET.CURVE:
'ESTABLISH SOURCE REGISTER AND 'ENCODING,ENSURE
'HEADER IS TURNED ON
WRT$ = "WFMPRE WFID:D,ENCDG:ASCII;HDR ON"
       CALL IBWRT(BD%,WRT$)
'RESERVE SPACE FOR THE DATA
    RD$ = SPACE$(2056)
'RETURN THE CURVE DATA
    WRT$ = "CURVE?"
       CALL IBWRT(BD%,WRT$)
       CALL IBRD(BD%,RD$)
'TRIM AND DISPLAY THE DATA ON SCREEN
    PRINT MID$(RD$,1,IBCNT%)
RETURN
    :
    PUT.CURVE:
'ESTABLISH ENCODING AND DESTINATION REGISTER
'SAVE THE REGISTER
WRT$ ="WFMPRE WFID:A,ENCDG:ASCII;SAVE A:ON"
       CALL IBWRT(BD%,WRT$)
'TRIM CURVE DATA AND SEND IT
'CURVE COMMAND HEADER IS INCLUDED IN RD$
    WRT$ = MID$(RD$,1,IBCNT%)
       CALL IBWRT(BD%,WRT$)
RETURN
```

## Transferring Files

The FILE command/query transfers named data files between the spectrum analyzer and controller. FILE and FILE? enables data from one instrument to be returned for disk storage and subsequent transmission to another instrument, or perhaps, to the same instrument in the event of NVRAM failure. They are not intended or well-suited for viewing curves or editing settings.

Permissible file names are established by the spectrum analyzer, and are listed under the FILE command discussion in Section 4, ***Command and Query Definitions***. The files are created within the spectrum analyzer's memory only as required. That is, a BSET03 file exists only if the B-register settings have been previously saved in the third storage location. The currently created files can be viewed by pressing the key sequence [UTIL MENU] [4] [6].





**Example 6-3. Subroutines to Return or Transmit Data Files.**

```
    GET.FILE:
'ENTER THE 2712 FILENAME FOR THE FILE TO UPLOAD
'SEE TABLE 4-1 FOR A LIST OF THE NAMES
    FILE2712$ = <2712 FILENAME>
'TURN ON HEADER, REQUEST FILE
    WRT$ = "HDR ON;FILE? "+FILE2712$
       CALL IBWRT(BD%,WRT$)
'READ FILE INTO STRING VARIABLE RD$
       CALL IBRD(BD%,RD$)
'TRIM STRING VARIABLE TO NUMBER
'OF CHARACTERS TRANSFERRED
    FILEDAT$ = MID$(RD$,1,IBCNT%)
    RETURN
       :
    PUT.FILE:
'SEND THE FILE IN MEMORY BACK TO THE 2712
    WRT$ = FILEDAT$
       CALL IBWRT(BD%,WRT$)
    RETURN
```

Example 6-3 lists QuickBASIC subroutines that can be used with the National Instruments GPIB board and software to transmit or return data files. You must supply the name of the file to be returned or transmitted.

The response header is turned on when the file is returned to the controller. This is how the returned data string (FILEDAT$) will appear:

```
FILEDAT$ = FILE "<filename>",<data block>
```

This is exactly the form required for the FILE command, so upon transmission you need only send FILEDAT$. Alternately, you can set HDR OFF in GET.FILE and then set WRT$ = "FILE "+FILEDAT$ in PUT.FILE.

Similar files may be sent to different locations, but you cannot transfer one type of file to another type. That is, you can return BSET04 and send it to CSET05, but you cannot return ACF2 and rename it UDP07.

The file name is embedded in the response, so this subroutine transmits the file to the same location that it came from. This limitation can be circumvented by inserting this line at the beginning of PUT.FILE:

```
MID$(FILEDAT$,1) = "FILE <new filename>"
```

These subroutines do not store or retrieve the files to and from disk. This can be done with the usual BASIC OPEN, PRINT # and INPUT # statements. Alternately, you can use the National Instruments IBWRTF() and IBRDF() calls to transfer data directly between the spectrum analyzer and disk. See Example 6-3 for an example of this approach.

## Plotting Spectrum Analyzer Screen Data

The PLOT? query enables the transfer of data representing an image of the 2711 and 2712 display screen from the spectrum analyzer to a plotter or printer. It performs a function similar to the front panel [PLOT] key. The printer or plotter must speak the HPGL language or be compatible with Epson FX codes, and the appropriate printer type must be specified either locally or using the PTYpe command.

It is possible for the spectrum analyzer to send data directly to a plotter. To do this, the GPIB ATN line must be held high while the spectrum analyzer is addressed as a talker and the plotter as a listener, and then ATN is set low. EOS ON and WAIt for an end-of-sweep SRQ must also be set. This approach requires no computer memory.

An alternate approach to plotting the spectrum analyzer screen data enables you to create independent input and output subroutines for use with specific devices, and to share those devices with other instruments on the GPIB. This approach involves returning the screen plot data to the controller, then sending it to the designated output device. The program does not proceed until the plot data are received, so it is not necessary to WAIt for an end-of-sweep SRQ. Further, this approach enables you to do the printing or plotting at a more convenient time or use an entirely different controller and output device.

Figures 6-1 and 6-2 show how you might return and print or plot instrument data. Each "Do it" in the diagrams can represent a separate subroutine for a specific device.





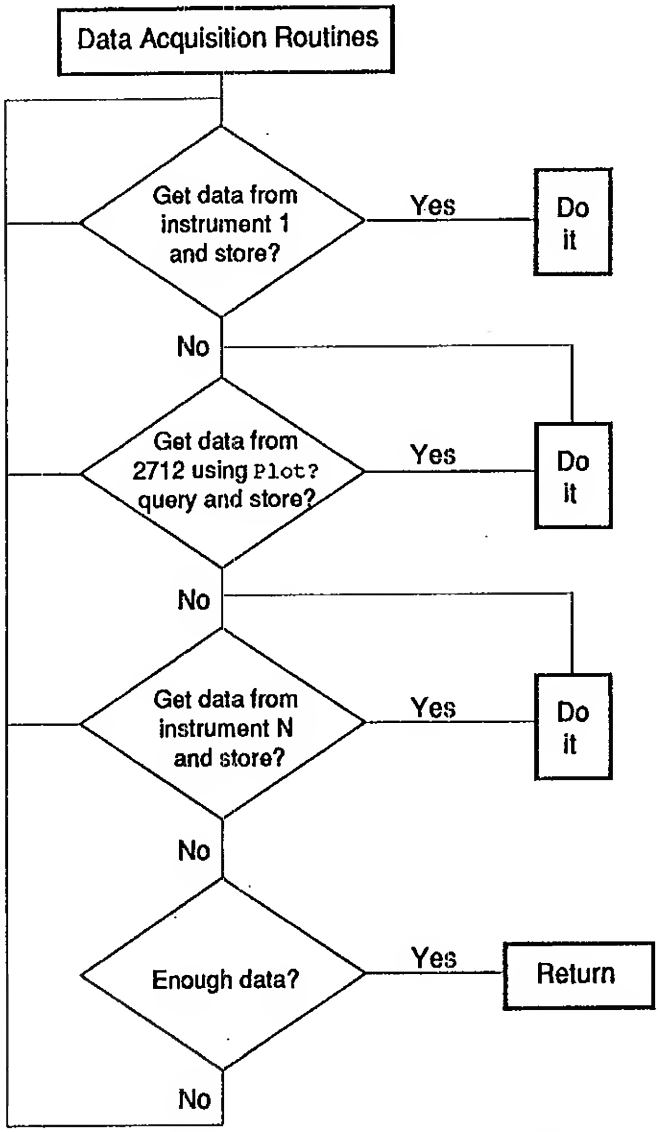



**Figure 6-1. Possible Data Acquisition Scheme.**

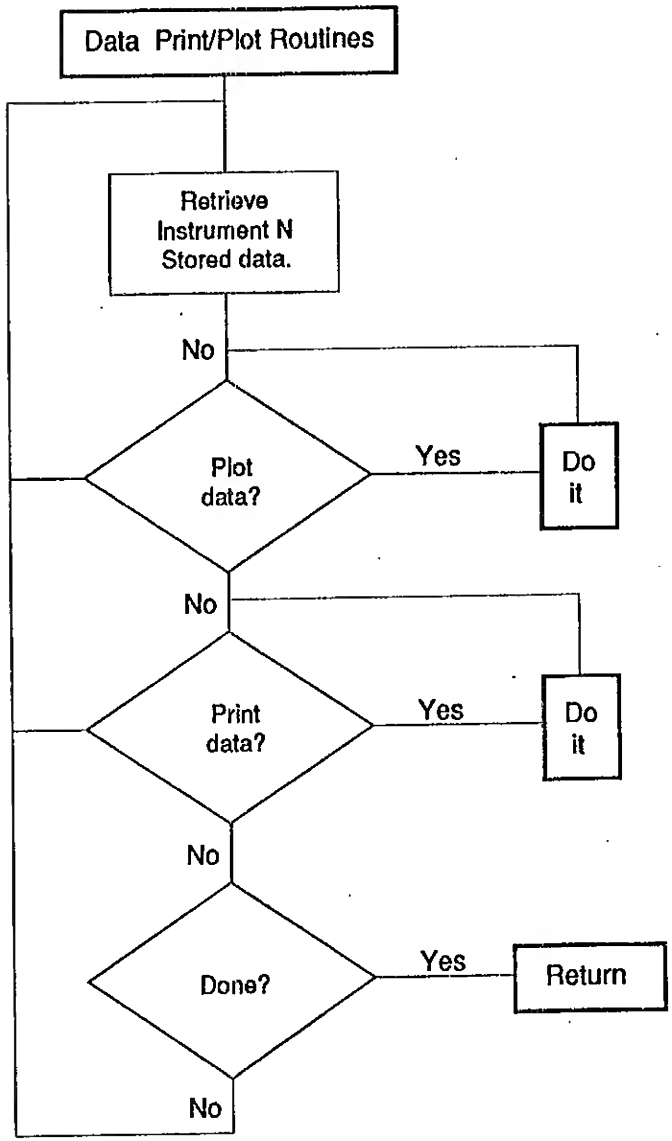



**Figure 6-2. Possible Data Print/Plot Scheme.**





Example 6-4 shows a subroutine for obtaining screen plot data from the 2712 using the PLOT? query. It does not store the data on disk (you can add that capability if desired), but holds them in memory as the string variable PLOT.DAT$.

The length of the string required depends on the display acquisition mode, the plotter type, and the number of registers displayed. 12 kbyte of memory is enough for a single sweep in PEAK acquisition mode if an HPGL plotter is used. If an Epson FX printer is used, as many as 61.1 kbyte may be required for four sweeps in MAX/MIN mode. Substitute 61100 for 12000 and change HPGL4 to EPSON if using an Epson FX printer.

Specify HPGL2 if you have a 2-pen plotter. Data may be sent to a parallel Epson-type printer on the controller's parallel port, but to send it over the GPIB, the printer must be equipped with a GPIB board (an unlikely but possible configuration).

The SEND.PLOT subroutine transmits PLOT.DAT$ to an HPGL 4-pen plotter matching the type specified in GET.PLOT. Note that the controller timeout is disabled, and execution following the plot is restarted by pressing any key.

## Returning the On-screen Readouts

The spectrum analyzer's on-screen readouts provide a summary of the important operational instrument parameters. The PRDouts? query enables returns most of these readouts to the computer. This query does not return the spectrum analyzer's general purpose message line, the GPIB status line, or the user-defined "DISPLAY MESSAGE" line.

The PRDouts? query does return the PRDOUTS header (if HDR is ON) and up to 14 arguments, depending on the spectrum analyzer's mode of operation and its current status. The arguments are listed in Section 4, ***Command and Query Definitions***, under PRDouts?. The response is ASCII-encoded and can be read and displayed on the controller screen using the subroutine in Example 6-5.

**Example 6-4. Subroutines to Return or Send Screen Plot Data.**

```
   GET.PLOT:
'RESERVE SPACE FOR SCREEN DATA
      PLOT.DAT$ = SPACE$(12000)
'SET PLOTTER TYPE AND REQUEST SCREEN DATA
      WRT$ = "PTYPE HPGL4;PLOT?"
         CALL IBWRT (BD%,WRT$)
'GET SCREEN DATA
         CALL IBRD (BD%,PLOT.DAT$)
'TRIM DATA TO NUMBER OF BYTES TRANSFERRED
      PLOT.DAT$=MID$(PLOT.DAT$,1,IBCNT%)
   RETURN
      :
   SEND.PLOT:
'DISABLES TIME OUT TO GIVE PLOTTER TIME TO FINISH
         CALL IBTMO(BD%,0)
      CLS
'SEND SCREEN DATA TO PLOTTER
         PLOTTER$ = "HC100"
         CALL IBFIND(PL%,PLOTTER$)
         CALL IBWRT(PL%,PLOT.DAT$)
'PRESS A KEY AFTER PLOTTER FINISHES
      PRINT "PRESS ANY KEY TO CONTINUE"
      DO WHILE INKEY$ = ""
         LOOP
'REESTABLISHES 30-SECOND TIME OUT
         CALL IBTMO (BD%,14)
   RETURN
```

**Example 6-5. Subroutine for Returning On-screen Readouts.**

```
   READOUTS:
'RESERVE SPACE FOR THE READOUTS
      READOUT.DAT$ = SPACE$(448)
'REQUEST THE READOUTS
      WRT$ = "PRDOUTS?"
         CALL IBWRT (BD%,WRT$)
'RETURN THE READOUTS
         CALL IBRD (BD%,READOUT.DAT$)
'TRIM DATA TO NUMBER OF BYTES RETURNED
      READOUT.DAT$ = MID$(READOUT.DAT$,1,IBCNT%)
'DISPLAY THE READOUTS
      PRINT READOUT.DAT$
   RETURN
```





## Saving and Restoring Equipment Settings

The SET? query can be used to return the current spectrum analyzer control settings. The settings are returned in a message format containing the command headers and arguments necessary to place the spectrum analyzer in its current configuration.

Settings can be saved in a controller disk file for later transfer to the same spectrum analyzer, or to another spectrum analyzer, when restoring the same operating environment. Because the SET header is always suppressed in the response, the response can be retransmitted as received.

The SET? query is generally used in preference to settings file transfers for several reasons:

- The same command and format can be used with a variety of Tektronix instruments for the same purpose.
- The 2711 and 2712 implement the retransmitted settings as soon as received rather than requiring a separate RECall command.
- The returned settings are in ASCII format and can be easily read if necessary.

See the SET? query in Section 4, ***Command and Query Definitions***, for details.

Example 6-6 shows QuickBASIC subroutines for storing and reestablishing the settings group. The routine is suitable for use with the National Instruments GPIB board and software.

**NOTE**

*You must substitute the disk file name in which settings are to be stored in place of "filename".*

### Example 6-6. Subroutines to Save and Restore Settings Groups.

```
   GET.SET:
'REQUEST THE SETTINGS
       WRT$ = "SET?"
           CALL IBWRT (BD%,WRT$)
'READ THE SETTINGS FROM THE 2712
           CALL IBRD (BD%,RD$)
'TRIM THE SETTINGS TO THE NUMBER OF BYTES RETURNED
       SETTINGS$ = MID$(RD$,1,IBCNT%)
'DISPLAY THE SETTINGS FOR VERIFICATION PURPOSES
       PRINT SETTINGS$
'SAVE SETTINGS ON DISK --SUBSTITUTE DISKFILE NAME
'THAT YOU WANT TO STORE SETTINGS UNDER FOR FILENAME
               OPEN "O",#1,"FILENAME"
       PRINT #1, SETTINGS$
       CLOSE #1
   RETURN
          :
   PUT.SET:
'OPEN THE DISK FILE AND READ IN THE STORED SETTINGS
       OPEN "I",#1,FILENAME$
       INPUT #1,SETTINGS$
       CLOSE #1
'DISPLAY THE SETTINGS FOR VERIFICATION PURPOSES
       PRINT SETTINGS$
'SEND THE SETTINGS TO THE 2712
       WRT$ = SETTINGS$
           CALL IBWRT (BD%,WRT$)
   RETURN
```





## Waiting For Results

A number spectrum analyzer functions require a wait period between the time they are requested or begin execution, and the time at which the results of the operation are available. These functions are listed below:

- Delta marker readouts
- Counter readouts
- Signal searches
- User-Definable Programs
- Marker readouts
- Ensemble averages
- Plots
- Normalizations

In Example 6-7 we have used the M,C MINUS SAVE A mode as an example of how to program for these events. This is only one example of many different possibilities when the `WAIt` command can be used. The WAit command is typically used in conjunction with the single sweep mode to maintain synchronization between the controller and spectrum analyzer during program execution.

### Example 6-7. Subroutine to Demonstrate the `WAIt` Command.

```
'================================================
' program segment to demonstrate the use of the
WAIT command
'================================================
'
'=======================
' set up for auto polling
'=======================
'
PEN OFF
ON PEN GOSUB serial.poll
PEN ON
'
'================================================
'use only waveform A and place instrument in single
sweep mode
'================================================
'
wrt$ = "VIEW A:ON;VIEW B:OFF;VIEW C:OFF;VIEW
D:OFF;SIGSWP"
CALL IBWRT(bd%, wrt$)
'
```

```
'================================
'turn on end of sweep indicator,
'trigger a single sweep,
'and WAIT until end of sweep
'================================
'
wrt$ = "EOS ON;SIGSWP;WAIT"
CALL IBWRT(bd%, wrt$)
'
'================================================
' halt program execution until EOS is indicated
' this guarantees that what is saved in A register
is correct
'================================================
'
END.OF.SWEEP$ = "N"
DO WHILE END.OF.SWEEP$ = "N"
LOOP
'
'================================================
' DO NOT turn on Save A until at least one sweep
has been made
'================================================
'
wrt$ = "SAVE A:ON"
CALL IBWRT(bd%, wrt$)
'
'================================================
' now use the waveform saved in A to subtract from
what is displayed in C
'================================================
'
wrt$ = "VIEW A:ON;VIEW B:OFF;VIEW C:ON;VIEW
D:OFF;VIEW MINUSA:ON;"
CALL IBWRT(bd%, wrt$)
'
wrt$ = "SIGSWP;WAIT"
CALL IBWRT(bd%, wrt$)
'
'================================================
'  wait here until end of sweep before continuing
'================================================
'
END.OF.SWEEP$ = "N"
DO WHILE END.OF.SWEEP$ = "N"
LOOP
'
```





```
'================================================
'  now reset the analyzer and view the C register
which contains the
'  difference between what is saved in A and what
is actively displayed
'================================================
'
wrt$ = "EOS OFF;VIEW A:OFF;VIEW B:OFF;VIEW
C:ON;VIEW D:OFF;VIEW MINUSA:ON;"
CALL IBWRT(bd%, wrt$)
'
'========================
' terminate auto polling
'========================
'
PEN OFF
'
'================
' exit module
'================
'
END SUB
```

## SAMPLE GPIB CONTROLLER

The following COMM2712 program is a simple utility for communicating with the 2711 or 2712 spectrum analyzer over the GPIB. It contains some of the subroutines (or elements of them) discussed earlier in this section, in addition to some new material. This program is not very sophisticated, but it does show how to command and interrogate the spectrum analyzer in a manner that enables you to perform several useful operations.

**Example 6-8. Sample GPIB Controller Program.**

```
'                COMM2712
'
'program to communicate with a Tektronix
'2712 Spectrum Analyzer via the GPIB
'
'declare GPIB system software reserved variables
   REM $INCLUDE: 'qbdecl4.bas'
'
'declare common global variables
   COMMON SHARED bd%,BDNAME$,RD$,wrt$
   COMMON SHARED event.code$,NUMBYT%
'
'dimension max size of returned data string
   RD$ = SPACE$(5000)
'
'dimension an integer array for packed
'integer CURve? response
   DIM SHARED CUR%(512)
'
'obtain bus device unit descriptor (BD%).
'BDNAME$ must match name
'established for the 2712 with the IBCONF program
   BDNAME$ = "TEK_SA"
      CALL IBFIND(BDNAME$, bd%)
'
'establish link to abnormal event handler;
'enable interrupt line
   ON PEN GOSUB ABNORM.EVE
   PEN ON
'
'enables SRQ generation in case of abnormal event
      CALL ibwrt(bd%, "RQS ON")

'trap DOS errors
   ON ERROR GOTO ERR.TRAP
```





```
'
'generate the menu
MENU:
   CLS
   PRINT "F1    SEND COMMAND OR QUERY"
   PRINT
   PRINT "F2    SAVE CURRENT SETTINGS TO FILE"
   PRINT
   PRINT "F3    RESTORE SETTINGS"
   PRINT
   PRINT "F4    SAVE AN INSTRUMENT FILE"
   PRINT
   PRINT "F5    RESTORE AN INSTRUMENT FILE"
   PRINT
   PRINT "F6    ACQUIRE CURVE DATA"
   PRINT
   PRINT "F10   EXIT"
   PRINT
   PRINT "PRESS F1-F6 OR F10 TO MAKE SELECTION"
   PRINT
'
'chk keyboard for keypress, decode,
'and branch to correct subroutine
KYBD.CHK:
   IN$ = INKEY$
   IF IN$ = "" GOTO KYBD.CHK
   IF LEN(IN$) = 1 THEN BEEP: GOTO KYBD.CHK
   IN.KEY = ASC(MID$(IN$, LEN(IN$), 1))
'decoding complete
      SELECT CASE IN.KEY        'branch to subroutine
         CASE 59                'F1 pressed
            GOSUB SEND.RCV
            GOTO MENU
         CASE 60                'F2 pressed
            GOSUB SAVE.SET
            GOTO MENU
         CASE 61                'F3 pressed
            GOSUB RES.SET
            GOTO MENU
         CASE 62                'F4 pressed
            GOSUB SAVE.FILE
            GOTO MENU
         CASE 63                'F5 pressed
            GOSUB RES.FILE
            GOTO MENU
         CASE 64                'F6 pressed
            GOSUB INT.CUR
```

```
            GOTO MENU
         CASE 68              'F10 pressed
            SYSTEM            'returns to dos
      END SELECT
   GOTO MENU                  'regenerates the menu
'
'subroutine to send a command or query
'and receive the response
SEND.RCV:
   CLS
   PRINT : PRINT "ENTER MESSAGE TO SEND"
   PRINT
   INPUT wrt$
      CALL ibwrt(bd%, wrt$)
      GOSUB GPIB.ERR
   hold.time = TIMER     'slight delay; srq check
   DO WHILE TIMER < hold.time + 1
      LOOP
   QUES = INSTR(1, wrt$, "?")
'if ques=0,there is no response;
   IF QUES = 0 THEN GOTO SEND.RCV
'
'if message contains "?" get response and print it
   CALL IBRD(bd%, RD$)
   GOSUB GPIB.ERR
   PRINT : PRINT "THE RESPONSE IS:"
   PRINT : PRINT MID$(RD$, 1, IBCNT%)
   PRINT : PRINT
   INPUT "SEND MORE (ENTER Y OR N)?"; Y$
   IF Y$ = "Y" THEN GOTO SEND.RCV
RETURN
'
'subroutine to fetch current instrument settings
'from 2712 and save to disk
SAVE.SET:
   CLS : PRINT
   PRINT "ENTER  NAME FOR SETTINGS FILE"
   PRINT "USE PATH IF NOT IN CURRENT DIRECTORY"
   INPUT FILENAME$
   WRT$ = "SET?"
      CALL IBWRT(BD%, WRT$)   'request settings
   GOSUB GPIB.ERR
   CALL IBRD(bd%, RD$)        'read settings
   GOSUB GPIB.ERR
   SETTINGS$ = MID$(RD$, 1, IBCNT%)
'
'eliminate extra characters and print
```





```
   PRINT : PRINT SETTINGS$
   PRINT : PRINT
   INPUT "OK TO STORE (ENTER Y OR N)? "; Y$
   IF Y$ = "N" THEN RETURN     'store if everything
   OPEN "O", #1, FILENAME$, IBCNT%    'looks OK
   PRINT #1, SETTINGS$
   CLOSE #1
RETURN
'subroutine to restore a group of instrument
'settings from disk to the 2712
RES.SET:
   CLS : PRINT
   PRINT "ENTER NAME OF SETTINGS FILE"
   PRINT
   INPUT FILENAME$
   OPEN "I", #1, FILENAME$     'read settings file
   INPUT #1, SETTINGS$
   CLOSE #1
   PRINT : PRINT SETTINGS$
   INPUT "OK TO RESTORE (ENTER Y OR N)? "; Y$
   IF Y$ = "N" THEN RETURN     'if displayed settings
   WRT$ = SETTINGS$            'OK, then restore
      CALL IBWRT(BD%, WRT$)    'to 2712
      GOSUB GPIB.ERR
RETURN
'
'subroutine fetches file from 2712, stores on disk
SAVE.FILE:
   CLS : PRINT
   PRINT "ENTER NAME OF 2712 FILE TO STORE"
   INPUT FILE2712$
'see FILE command for 2712 file names
   PRINT
   PRINT "ENTER NAME OF DISK FILE for STORing"
   INPUT FILENAME$
   FILE2712$ = UCASE$(FILE2712$)
   WRT$="HDR ON;FILE?
"+CHR$(34)+FILE2712$+CHR$(34)
      CALL IBWRT(BD%, WRT$)
'
'request file transfer
      GOSUB GPIB.ERR
      CALL IBRDF(bd%, FILENAME$)     'read and store
      GOSUB GPIB.ERR              '2712 file to disk
RETURN                            'as FILENAME$
'subroutine restores 2712 file from disk to the 2712
RES.FILE:
```

```
    CLS : PRINT
    PRINT "ENTER DISK FILE TO RESTORE TO 2712"
    INPUT FILENAME$              'note: the file named
        CALL IBWRTF(bd%, FILENAME$)    'FILENAME$
        GOSUB GPIB.ERR          'contains the name of
RETURN          'the 2712 file to be restored
'
'subroutine to fetch curve data in packed binary
'form and convert it to 2-byte integer format
INT.CUR:
    PRINT
    PRINT "GET CURVE FROM WHICH REGISTER?"
    INPUT "      (ENTER A, B, C, OR D)    "; REG$
    PRINT
'ensure response header is on
        CALL IBWRT(BD%, "HDR ON")
        GOSUB GPIB.ERR
'tell 2712 which register and encoding to use
    WRT$ = "WFMPRE WFID:" + REG$ + ",ENCDG:BINBLK"
        CALL IBWRT(BD%, WRT$)
        GOSUB GPIB.ERR
        CALL IBWRT(BD%, "CUR?")
        CALL IBRDI(BD%, CUR%(), 9)     'fetch curve
        GOSUB GPIB.ERR      'data in packed binary;
        CALL IBrdi(Bd%, Cur%(), 512)
'write over header characters (1st 9) with data
        CALL DEBLK(CUR%(), CUR%(), 512, 8, NUMBYT%)
        PRINT "# OF BYTES CONVERTED = "; NUMBYT%
RETURN
'DEBLK unpacks binary data and re-stores as 2-byte
'integers in same array. NUMBYT$ always equals 512
'subroutine to find and display event code
following 'an SRQ created by an abnormal event
ABNORM.EVE:
    CLS
    PRINT "AN ABNORMAL EVENT HAS OCCURRED."
    PRINT
        GOSUB SERIAL.POLL    'call subroutines to poll
        GOSUB EVENT.FIND     '2712 and find event code
    PRINT
    PRINT "R TO RESTART; ANY OTHER KEY TO END"
    INPUT KEY$
    'pressing R returns to
    IF KEY$ <> "R" THEN END     'menu; variables
        GOTO MENU               'are not erased
RETURN
'
```





```
'subroutine to serially poll the 2712,
'read the status byte, and print it
'used as part of abnormal event handler,
'but can be used any time to obtain status byte;
'see your GPIB documentation for more information
SERIAL.POLL:
'
'read and print status byte; reset SRQ
       CALL IBRSP(BD%, SPR%)
    PRINT "STATUS BYTE = "; SPR%
RETURN
'
'Subroutine to find event code using the EVE? query;
'result valid only after serial poll if RQS is ON
EVENT.FIND:
    PRINT "EVENT CODE(S) IS:";
    WRT$ = "HDR OFF;EVE?"       'turn off header and
    event.code$ = SPACE$(5)     'request event code
        CALL IBWRT(BD%, WRT$)   'send command
        CALL IBRD(BD%, event.code$)    'read code
    PRINT event.code$                  'print code
        GOSUB GPIB.ERR
RETURN
'subroutine to print the GPIB error code
'if a GPIB error occurs, and
'end the program gracefully
GPIB.ERR:
    IF IBSTA% < 32768 THEN RETURN      'no GPIB error
    CLS : PRINT          'if GPIB status word<32768
    PRINT "GPIB ERROR HAS OCCURRED."
    PRINT : PRINT "GPIB ERROR CODE IS "; IBERR%
    PRINT : PRINT
    PRINT "CHECK YOUR SYSTEM AND RESTART."
    END
RETURN
'
'subroutine to end program gracefully on DOS error
ERR.TRAP:
    CLS : PRINT
    PRINT "DOS ERROR HAS OCCURRED."
    PRINT : PRINT
    PRINT "CHECK YOUR SYSTEM AND RESTART."
    END
RESUME NEXT
```

# REMOTE MENU CONTROL

The 2711 and 2712 spectrum analyzers include a set of commands and queries for designing menus on the spectrum analyzer screen and interacting with these menus by pressing keys on the spectrum analyzer KEYPAD. This feature is designed primarily for making automated measurements with the Tektronix 2402A Tekmate, an external, portable, PC-based controller without a display or keyboard. These features can also be used with a standard PC controller to define a menu remotely. This allows the spectrum analyzer user to interact with menu functions locally, at the spectrum analyzer's KEYPAD, and see results on the spectrum analyzer's display.

Remotely-defined menus are controlled by these four commands:

- DEFMenu: defines menu lines to print on the display
- CLRMenu: clears the current menu from the display
- KEY?: returns the identity of the last key pressed
- CLRKey: clears pending key presses

The syntax and use of these commands, including side effects, are described in Section 4, ***Command and Query Definitions***. Figure 6-3 shows a test menu that could be defined by a user. This example shows how different types of actions can be implemented by selecting the menu commands from a user-defined menu.

The following sequence of instrument-specific commands are used to create the menu of Figure 6-3:

```
CLRMENU
DEFMENU L1:"TEST MENU"
DEFMENU L3:" 0 INIT INSTRUMENT"
DEFMENU L4:" 1 CHANGE REF LEVEL      (NOW 12.3DBM)"
DEFMENU L5:" 2 PERFORMANCE TESTS"
DEFMENU L16:"      ""<-"" = PREVIOUS MENU"
```

The initial CLRMenu command clears every line in the menu so only the lines that contain information must be defined. This command also clears the last key pressed so that subsequent KEY? queries return NULL until the user presses a key. Any key presses prior to the menu definition are discarded.





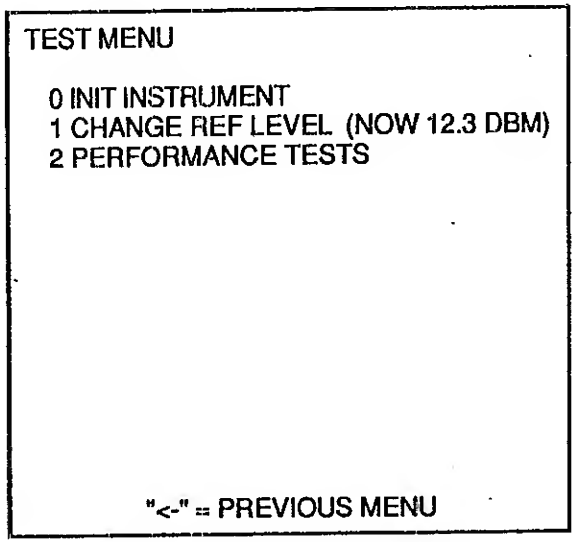



**Figure 6-3. A Remote Menu.**

Notice the use of paired quotes ("") to represent a single quote mark on the menu display. This menu could also have been defined using a single DEFMenu command with arguments separated by commas in place of the four individual DEFMenu commands. Refer to Section 4, ***Command and Query Definition***, for the syntax of this command.

## Execute and Exit

Item 0 on the remote menu of Figure 6-3 is the simplest type of entry. The desired action is instrument initialization and an exit from the menu when the operator presses the [0] key on the spectrum analyzer's front panel KEYPAD. This is accomplished by the following algorithm executing on either the Tektronix 2402ATekmate or a controller:

- Do KEY? queries until the result is not NULL.
- If "M0" is returned (key [0] has been pressed), execute an INIT followed by a CLRMenu command.

This is a very simple example. The 2402A Tekmate (or your PC controller) is capable of performing a complex test sequence in response to a single keystroke.

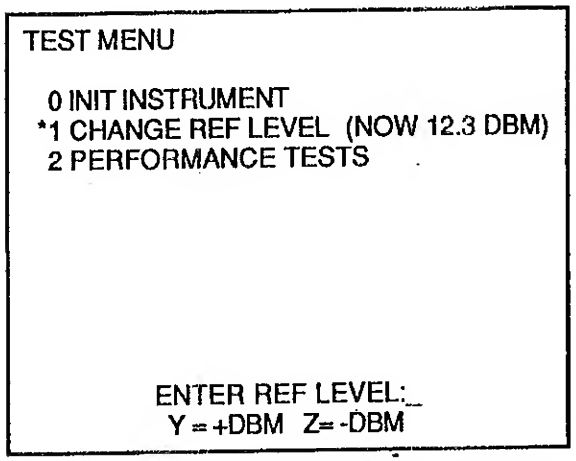



**Figure 6-4. Prompting for a Numeric Entry.**

## Numeric Entry

Item 1 on the remote menu of Figure 6-4 is a numeric entry item. This is a more complex entry because it requires additional interaction with the operator. When the spectrum analyzer's front panel [1] key is pressed, a numeric entry line is displayed at the bottom of the screen with some terminator keys (Figure 6-4). The user enters a number and terminator from the spectrum analyzer KEYPAD in response to this prompt.

The terminator key signals the 2402A Tekmate (or PC) when the numeric entry is finished, and also specifies the units. The 2402A Tekmate then programs the value into the instrument, or it may use the entry as a parameter for itself, as outlined in the following algorithm:

- Do KEY? queries until the result is not NULL.
- If "M1" is returned (key [1] has been pressed), execute the following commands:

```
DEFMENU L4:" *1 CHANGE REF LEVEL      (NOW 12.3DBM)"
DEFMENU L15:"      ENTER REF LEVEL:_"
DEFMENU L16:"      Y = +DBM  Z = -DBM"
```





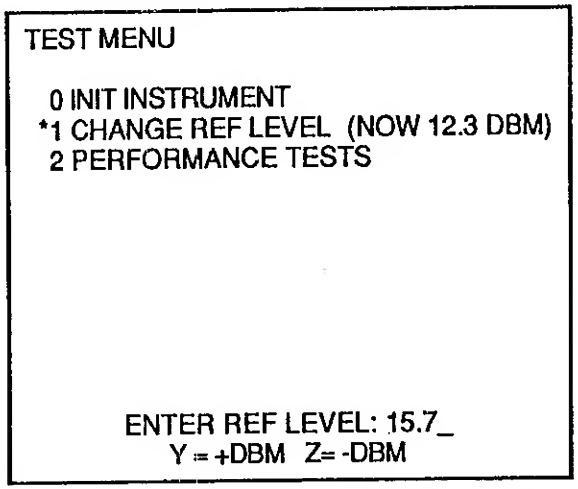



**Figure 6-5. Specifying a Numeric Value.**

Next the programmer should use an algorithm similar to the following one to accept the data entry:

- Do KEY? queries until the result is not NULL.
- Use appropriate error checking to ensure only numeric, decimal point, and terminator key entries are accepted.
- Begin building an ASCII string by appending successive keypresses to a string variable, say VAL$.
- Change line 15 of the menu with this command:

```
DEFMENU L15:"    ENTER REF LEVEL: " + VAL$ + "_"
```

- Loop until the whole number has been entered and a terminator has been detected.

For example, if the keys [1] [5] [.] and [7] are pressed, the menu display will change as shown in Figure 6-5. When a terminator key is pressed, the program must respond by setting the reference level using the REF command:

```
REF 15.7 DBM
CLRMENU
```

The CLRMENU command automatically exits the menu. If desired, you may remain in the remote menu by issuing the necessary DEFMENU commands to rewrite the screen.

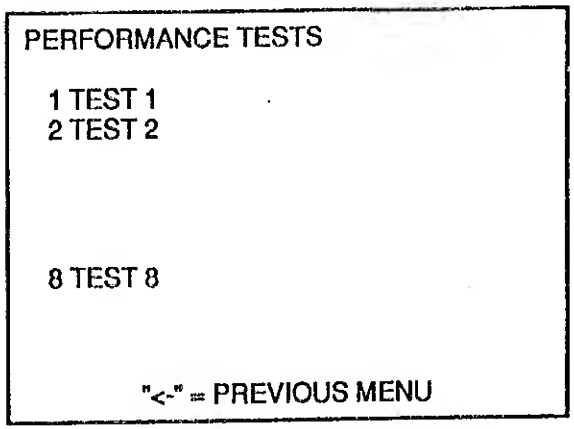



**Figure 6-6. A Remote Submenu.**

## Defining a Submenu

Item 2 on the remote menu of Figure 6-5 shows how submenus might be created. Use the 2402A Tekmate to execute the following algorithm for an example of building a submenu:

- Do KEY? queries until the result is not NULL.
- If the key is "M2", send the following commands:

```
CLRMENU
DEFMENU L1:"PERFORMANCE TESTS"
DEFMENU L4:" 1 TEST 1"
DEFMENU L5:" 2 TEST 2"
DEFMENU L9:" 8 TEST 8"
DEFMENU L16:"     ""<-"" = PREVIOUS MENU"
```

The resulting menu is shown in Figure 6-6. Note that the spectrum analyzer does not understand a nested menu structure. The nesting structure is maintained by the application program running on the controller.

The application program also implements the Backspace [BKSP] key to return to a higher menu level, as with the standard 2711 and 2712 menus.





# SAMPLE RS-232 CONTROLLER

The program in Example 6-9 illustrates interactive control of the 2711 or 2712 spectrum analyzer via the RS-232 interface. The program can be revised for GPIB by modifying the device-dependent initialization, error handling, and I/O functions contained in the main module subroutine READ.BUFFER, and in the RS232.CALLS module.

Example 6-9 configures the 2712's RS-232 interface to use the VERBOSE mode (see Section 1, ***Introduction***), causing the instrument to send a response back to the program for each command it receives. The program terminates normally if the operator enters [ESC] from the computer keyboard. Otherwise, the program analyzes and displays each possible VERBOSE mode response and takes one of the following actions:

- If a command results in response "OK", the program displays the response and waits for the next operator entry.
- If a command or query results in an error, the program retrieves and displays the 2712's response, "ERR," followed by the associated event code. It then awaits the next operator entry.
- If a query does not result in the response "ERR," the program displays the spectrum analyzer's query response and awaits the next operator entry.

This interactive program handles all instrument functions and includes extensive error checking and a friendly human interface. Program variables and their use, as well as subroutine linkage, are clearly defined and commented in the code. The program declares the following procedures:

ENTERCOMMAND

This procedure formats the computer's display screen to accept operator input and calls SENDCOMMAND. The call to this routine can be replaced by a call to any user-written module, for example, to perform a full harmonic distortion analysis. (All RS-232 communications are already in place.)

SENDCOMMAND

This procedure sends commands to the spectrum analyzer and receives responses from the spectrum

analyzer after establishing communications over the RS-232 interface. It also calls RS232.CALLS to perform actual RS-232 communications.

PAUSE

This procedure causes the controller to pause a specified number of seconds. It is used during initialization to prevent interference with the spectrum analyzer's configuration. (See the RS232 command in Section 4, ***Command and Query Definitions.***)

RS232.CALLS

This procedure performs all input/output for RS-232 communications between the controller and the spectrum analyzer. It also checks for communication errors and displays error messages.

The RS-232 sample program performs the following functions:

1. Opens COM1 and establishes RS-232 communications with the spectrum analyzer using this configuration:

| | |
|---|---|
| Baud rate: | 9600 |
| Data bits: | 8 |
| Stop bits: | 1 |
| Parity: | None |
| Verbose: | ON |
| Echo mode: | OFF |
| Terminator: | CR LF |
| Flow control: | NONE |

   If the spectrum analyzer is not properly configured the program displays an error message. Refer to the description of the RS232 command in Section 4, ***Command and Query Definitions***, and the RS-232 installation procedure in Section 1, ***Installation***, for more information on RS-232 configuration.

2. It accepts a spectrum analyzer command that the operator enters at the computer keyboard and sends the command to the spectrum analyzer. The program terminates when the operator presses the [ESC] key on the computer keyboard.

3. In all cases the program displays the spectrum analyzer's VERBOSE response to the command or query and awaits further operator input.





**Example 6-9. Sample RS-232 Controller Program.**

```
'************************************************
' This is an example of an interactive program
' allowing an operator to communicate with the 2712
' spectrum analyzer.  Commands can be sent to the
' 2712 and responses are read back.
' All communication is via RS232 and is contained
' in the procedure RS232.CALLS and in the
' subroutine READ.BUFFER.  Communications are
' assumed to be passed through COM PORT 1.
'************************************************
' Communications are established with a baud rate of
' 9600, data bits set to 8, parity of NONE, EOL=CRLF
' FLOW CONTROL = NONE, ECHO = OFF, and VERBOSE=ON.
'
'************************************************
'
' **** PROGRAM EXPECTS 2712 TO BE SET THIS WAY *****
'
' If the 2712 is not so configured, an error message
' is displayed directing the operator to configure
' the 2712 properly.
'
'************************************************
DECLARE SUB SENDCOMMAND ()
DECLARE SUB ENTERCOMMAND ()
DECLARE SUB PAUSE (p%)
DECLARE SUB RS232.CALLS ()
'
'================================================
' global variables used for communications linkage
'================================================
'
COMMON SHARED rd$, wrt$, func%, errflg$,
end.of.read$, buffer$, wfm%()
'
'=======================================
' ARRAY to hold waveform acquired from 2712
'=======================================
'
DIM SHARED wfm%(511)
'
'------------------------------
'begin program execution here -
'------------------------------
'
```

```
BEGIN.PROGRAM:
'
'======================
' set up the ESCape key
'======================
'
KEY(15) OFF
KEY 15, CHR$(&H0) + CHR$(&H1)
ON KEY(15) GOSUB END.PROGRAM
KEY(15) ON
'
'==============================
' set up the communications key
'==============================
'
ON COM(1) GOSUB READ.BUFFER
'=========================================
' now execute routine which will accept all
' user input and display all responses.
' procedure continues until the user
' presses the ESCape key.
'=========================================
'
CALL ENTERCOMMAND
'
'================================================
'end program and return to DOS when ESC is pressed
'================================================
'
END.PROGRAM:
'
COLOR 7, 0
CLS
END
'
'************************************************
'*                                              *
'* read routine for RS232                       *
'* branched from RS232.CALLS subprogram         *
'*                                              *
'************************************************
'
'===========
READ.BUFFER:
'===========
'
COM(1) OFF              'turn automatic branch off
```





```
'
'initialize string to save response from 2712
rd$ = ""
'
'set time limit for reading response from 2712
'binary waveform transfer may take longer
'ten seconds will be enough for others
'normalize query takes longer to respond
IF INSTR(wrt$, "NORM") THEN
   hold.i = 40
ELSEIF func% = 5 THEN
   hold.i = 15
ELSE
   hold.i = 10
END IF
'
'initialize counter and
'try to read for time allotted to make sure
'controller is just not too fast for the 2712
i = TIMER
DO WHILE TIMER < i + hold.i
   GOSUB READ.INPUT
   IF end.of.read$ = "Y" THEN
      i = 0
   END IF
LOOP
'
'set flag to avoid possible endless loop
'in RS232 calls
end.of.read$ = "Y"
RETURN
'
'==========
READ.INPUT:
'==========
'
'initialize flag to indicate that read something
buffer$ = "N"
'
'read entire contents of buffer; accumulate
response
'and set buffer flag on so know have read something
DO WHILE NOT EOF(1)
   a$ = INPUT$(LOC(1), #1)
   rd$ = rd$ + a$
   buffer$ = "Y"
LOOP
```

```
'************************************************
'*                                              *
'* following code segment distinguishes between a *
'* binary waveform transfer and any other response *
'* from the 2712.  Binary transfers can contain  *
'* embedded cr/lf characters; must be ignored until*
'* we get past the binary data.                 *
'*                                              *
'************************************************
'
'use first cr/lf encounter for eof on
'of everything but binary files
'ensure that EOR (cr/lf) read
'set buffer flag on to indicate successful read
IF func% <> 5 THEN
    IF INSTR(rd$, CHR$(10)) THEN
        IF INSTR(rd$, CHR$(13)) THEN
            end.of.read$ = "Y"
            buffer$ = "Y"
        END IF
    END IF
'ensure to have read past binary data
'before checking for cr/lf, set buffer flag on
ELSE
    IF LEN(rd$) > 523 THEN
        IF INSTR(524, rd$, CHR$(13)) THEN
            IF INSTR(525, rd$, CHR$(10)) THEN
                end.of.read$ = "Y"
                buffer$ = "Y"
            END IF
        END IF
    END IF
END IF
'
RETURN
'************************************************
'PROCEDURE to perform a number of tasks:        *
'1) format the screen to accept operator input  *
'2) open communication link with 2712           *
'   (func%=1,call RS232.CALLS)                  *
'3) accept input                                *
'4) link SENDCOMMAND which sends input to 2712  *
'                                               *
'The ESCape key ends program execution and      *
'returns operator to DOS.                       *
'************************************************
SUB ENTERCOMMAND
```





```
'
'================================================
' fill in the upper portion of screen with help info
'================================================
'
SCREEN 0
COLOR 0, 0
CLS
COLOR 12, 0
LOCATE 1, 26, 0
PRINT "2712 TALK/LISTEN DEMO FOR RS232";
COLOR 14, 9
FOR i% = 3 TO 8
    LOCATE i%, 6
    PRINT STRING$(70, " ");
NEXT
'
LOCATE 4, 15, 0
PRINT "THIS ROUTINE ACCEPTS SINGLE COMMANDS";
PRINT "OR QUERIES ONLY.";
LOCATE 6, 23, 0
PRINT "- Press the (Enter) key to send  ";
LOCATE 7, 23, 0
PRINT "- Press the (Esc) key to end ";
'====================================
' define a window for operator input
'====================================
'
COLOR 12, 0
LOCATE 9, 8, 0
PRINT "COMMAND INPUT";
COLOR 14, 9
FOR i% = 10 TO 17
    LOCATE i%, 6
    PRINT STRING$(70, " ");
NEXT
'
'=========================================
' define a window for instrument response
'=========================================
'
COLOR 12, 0
LOCATE 18, 8, 0
PRINT "SPECTRUM ANALYZER RESPONSE";
COLOR 14, 9
FOR i% = 19 TO 25
    LOCATE i%, 6
```

```
    PRINT STRING$(70, " ");
NEXT
'
'=======================
'set up linkage to 2712
'=======================
'
func% = 1
CALL RS232.CALLS
'
'=================================================
'initialize string which holds communication message
'=================================================
'
wrt$ = ""
'
'================================================
'now accept user input which will be sent to 2712
'================================================
'
LOCATE 11, 8, 1
'=======
INLOOP:
'=======
IN$ = INKEY$
IF IN$ = "" THEN
    GOTO INLOOP
ELSEIF LEN(IN$) <> 1 THEN
    BEEP
    GOTO INLOOP
ELSEIF IN$ = CHR$(27) THEN
    COLOR 7, 0
    CLS
    END
ELSEIF IN$ = CHR$(13) AND LEN(wrt$) = 0 THEN
'carriage return with no message is an error
    BEEP
    GOTO INLOOP
ELSEIF IN$ = CHR$(13) THEN      'cr signals to send
    CALL SENDCOMMAND    'to user input to 2712
    wrt$ = ""  'reinitialize string holding input
    FOR x% = 11 TO 16  'reinitialize screen input area
    LOCATE x%, 8: PRINT STRING$(66, " ");
    NEXT
    LOCATE 11, 8, 1             'reposition cursor
    GOTO INLOOP  'return to accept more input
ELSEIF IN$ = CHR$(8) AND LEN(wrt$) = 0 THEN
```





```
    BEEP
'backspace without a place to go is an error
    GOTO INLOOP
ELSEIF IN$ = CHR$(8) THEN
    GOSUB BACKSPACE
    GOTO INLOOP
ELSE
    wrt$ = wrt$ + IN$  'accumulate message from user
    x = POS(0): y = CSRLIN      'one byte at a time
    LOCATE y, x: PRINT IN$;     'and reposition cursor
    x = x + 1    'increment position of cursor
    IF x = 74 AND y = 16 THEN  'check cursor position
        FOR index% = 11 TO 16   'if outside window
            LOCATE index%, 8    'clear input window
            PRINT STRING$(66, " ");
        NEXT
        LOCATE 11, 8, 1         'and relocate cursor
        GOTO INLOOP
    ELSEIF x = 74 THEN          'if cursor beyond line
        CurRow% = CSRLIN        'move it down to next line
        LOCATE CurRow% + 1, 8, 1
        GOTO INLOOP
    ELSE
        GOTO INLOOP      'if cursor position ok just
    END IF               'return to accept more input
END IF
'
'*********************************************
' routine to back up cursor and erase character *
' this is a DESTRUCTIVE backspace routine       *
'*********************************************
'
'=========
BACKSPACE:
'=========
'
x = POS(0)
y = CSRLIN
'
IF x = 8 AND y = 11 THEN
    BEEP       'cannot backspace beyond area of window
    RETURN
ELSEIF x = 8 THEN
    y = y - 1    'backspace to end of previous line
    LOCATE y, 73        'and locate cursor
ELSE                              '
    LOCATE y, x - 1     'locate cursor 1 position back
```

```
END IF
'
PRINT " " + CHR$(29);   'erase character, back up
wrt$ = LEFT$(wrt$, LEN(wrt$) - 1)
'remove 1 character from string of saved input
'
RETURN
END SUB
'
'************************************************
' procedure to wait a given period of time.      *
' This pause routine only accepts integers since *
' less than a one second delay is not necessary. *
'************************************************
SUB PAUSE (p%)
i = TIMER
DO WHILE TIMER < i + p%
LOOP
END SUB
'
'************************************************
'* This procedure performs all RS232 calls. All  *
' information needed is defined in global variables*
'* rd$ = returned info from RS232 call           *
'* wrt$ = info to send to device                 *
'* func% = identifies RS232 function to perform  *
'* wfm%  = array to hold waveform                *
'************************************************
SUB RS232.CALLS
'
IF func% = 1 THEN
   GOTO SELECT.DEVICE
ELSEIF func% = 3 THEN
   GOTO SEND.MESSAGE.TO.DEVICE
ELSEIF func% = 5 THEN
   GOTO GET.BINARY.WAVEFORM
END IF
'************************************************
'* This routine opens COM PORT 1 for communication *
'* via RS232, opened for random to handle both   *
'* input and output.                             *
'************************************************
'* SPEED =  9600  (bits per second or BAUD RATE)  *
'* PARITY =  NONE  (parity bit not used)          *
'* DATA =  8  (number of data bits per byte)      *
'* STOP =  1  (number of stop bits)               *
'* EOL= CRLF (carriage return/line feed terminator)*
```





```
'* FLOW CONTROL = NONE  (no "handshaking")       *
'* ECHO = OFF (only needed for terminal emulation) *
'* VERBOSE = ON (responds for each communication)  *
'************************************************
'* If the 2712 has not been configured to agree   *
'* with the above parameters the program displays  *
'* an error message and forces the user to change  *
'* the 2712. Program will not execute otherwise.   *
'************************************************
'* CS = 0  (suppress checking CLEAR TO SEND line)  *
'* DS = 0 (suppress checking DATA SET READY line)  *
'* RB = 2048 (size in bytes of receive buffer)     *
'* TB = 2048 (size in bytes of transmit buffer)    *
'************************************************
'
SELECT.DEVICE:
'
buffer$ = "N"    'initialize these two indicators
read.error% = 0
'
OPEN "com1:9600,n,8,1,CS,DS,rb 2048,tb 2048"_
       FOR RANDOM AS #1
'
'================================================
' clear RS232 buffers and reset STATUS reporting
' to ensure no messages pending
'================================================
'
lc.status = INP(&H3FB)  'COM1 line control register
lc.status = lc.status OR 64    'set break bit to on
OUT &H3FB, lc.status    'send back modified
register
'
CALL PAUSE(1)           'wait one second
'
lc.status = INP(&H3FB)  'get COM1 line control reg
lc.status = lc.status AND (NOT 64)    'reset break
bit
OUT &H3FB, lc.status
'send back the modified register contents
'
================================================
' now continue verification of the com port set up
================================================
'
PRINT #1, "RS232 VERBOSE:ON"
GOSUB READ.FOR.VERBOSE
```

```
'
IF read.error% = 1 THEN
   GOTO SELECT.DEVICE
END IF
'
PRINT #1, "RS232 ECHO:OFF"
GOSUB READ.FOR.VERBOSE
'
IF read.error% = 1 THEN
   GOTO SELECT.DEVICE
END IF
'
PRINT #1, "RS232 EOL:CRLF"
GOSUB READ.FOR.VERBOSE
'
IF read.error% = 1 THEN
   GOTO SELECT.DEVICE
END IF
'
PRINT #1, "RS232 FLOW:NONE"
GOSUB READ.FOR.VERBOSE
'
IF read.error% = 1 THEN
   GOTO SELECT.DEVICE
END IF
' set up the 2712 so REQUEST message does not
' appear on analyzer and so the 2712 returns
' header information
'
PRINT #1, "RQS Off;HDR ON"
GOSUB READ.FOR.VERBOSE
'
EXIT SUB
'
'*************************************************
'* This routine sends the command string and reads *
'* the response                                    *
'*************************************************
'
SEND.MESSAGE.TO.DEVICE:
'
PRINT #1, wrt$
'
'=======================================
' begin read loop for response from 2712
'=======================================
'
```





```
GOSUB READ.FOR.VERBOSE
'
EXIT SUB
'
'*************************************************
'* This routine reads the 2712 response to get     *
'* a binary waveform.                              *
'*************************************************
'
GET.BINARY.WAVEFORM:
'
GOSUB READ.FOR.VERBOSE
'
IF LEN(rd$) >= 14 THEN
'strip off checksum ,semi-colon and cr/lf
'strip off header and byte count
'convert each binary value to ascii
   rd$ = LEFT$(rd$, LEN(rd$) - 4)
   rd$ = RIGHT$(rd$, LEN(rd$) - 9)
   FOR x% = 0 TO 511
      wfm%(x%) = ASC(MID$(rd$, x% + 1, 1))
   NEXT
END IF
EXIT SUB
'
'*************************************************
'* This routine displays a message if RS-232       *
'* communications cannot be established  .         *
'*************************************************
'
ERROR.DISPLAY:
'
'save cursor coordinates
'
hold.x% = POS(0)
hold.y% = CSRLIN
'=================
'save screen image
'=================
REDIM before$(7, 41), colr%(7, 41)
FOR p% = 0 TO 7
   FOR q% = 0 TO 41
   colr%(p%, q%) = SCREEN(p% + 18, q% + 20, 1)
   before$(p%, q%) = CHR$(SCREEN(p% + 18, q% +
20))
   NEXT q%
NEXT p%
```

```
'=========================================
' next print the window and the error message
'=========================================
COLOR 14, 6
FOR z% = 18 TO 25
    LOCATE z%, 20, 0
    PRINT STRING$(42, CHR$(32));
NEXT
 LOCATE 18, 30, 0
PRINT "COM(1) PORT PROBLEM";
LOCATE 19, 25
PRINT "Verify RS232 Port Configuration";
LOCATE 20, 31
PRINT "(UTIL 4/0/2)";
LOCATE 21, 22
PRINT "STATUS..... ONLINE  BAUD RATE..... 9600";
LOCATE 22, 22
PRINT "DATA BITS.. 8       PARITY........ NONE";
LOCATE 23, 22
PRINT "EOL........ CRLF    FLOW CONTROL.. NONE";
LOCATE 24, 22
PRINT "ECHO....... OFF     VERBOSE....... ON";
'
LOCATE 25, 27, 0
PRINT "Press (any key) to continue";
DO WHILE INKEY$ = ""
LOOP
'
'=======================
'redisplay original screen
'=======================
'
hold.min% = 0
max% = 41
min% = 0
FOR index% = 1 TO 8
    FOR p% = 0 TO 7
        FOR q% = min% TO max% STEP 7
            frgrnd% = colr%(p%, q%) MOD 16
            bckgrnd% = (((colr%(p%, q%) - frgrnd%)_
                                / 16) MOD 128)
            COLOR frgrnd%, bckgrnd%
            LOCATE p% + 18, q% + 20
            PRINT before$(p%, q%);
        NEXT q%
    NEXT p%
     IF min% > hold.min% THEN
```





```
        min% = min% + 1
        hold.min% = min%
    ELSE
        min% = min% + 1
        hold.min% = 0
    END IF
NEXT index%
'
LOCATE hold.y%, hold.x%
COLOR 14, 9
rd$ = ""
'
RETURN
'
'*************************************************
'* This routine performs all reading and error    *
'* checking. Uses 2712 VERBOSE mode.              *
'*************************************************
'
READ.FOR.VERBOSE:
'
'set two flags used in reading response
end.of.read$ = "N"
error.query$ = "N"
'enable event trapping for communication
'on communications port 1
COM(1) ON
i = TIMER
'allow 10 seconds to respond from event query
'if no response, assume communication not established
DO WHILE end.of.read$ = "N"
    IF TIMER > i + 10 THEN
        COM(1) OFF                      'set flags
        end.of.read$ = "Y"
        error.query$ = "Y"
    END IF
LOOP
'=================================================
' begin checking for error flags on return from read
'=================================================
'
IF error.query$ = "Y" THEN      'no communication open
    GOSUB ERROR.DISPLAY         'display error
    read.error% = 1             'set error flag
    CLOSE #1                    'close com port
ELSEIF buffer$ <> "Y" THEN
    GOSUB ERROR.DISPLAY         'do the same as above
```

```
    read.error% = 1              'set error flag
    CLOSE #1
END IF
'
RETURN
'
END SUB
'
'*************************************************
' procedure to SENDCOMMAND to the 2712 after input *
' from the ENTERCOMMAND procedure.                  *
' Command passed in the global string "WRT$"        *
' NOTE:                                             *
' Only SINGLE commands can be sent by the user.     *
'*************************************************
SUB SENDCOMMAND
'
wrt$ = UCASE$(wrt$)
'
FOR IX% = 20 TO 24       'clear out response window
    LOCATE IX%, 8
    PRINT STRING$(66, " ");
NEXT
'
LOCATE 20, 8                     'reposition cursor
'
POSIT1 = INSTR(wrt$, "?")        'all queries end with "?"
'
IF POSIT1 = 0 THEN               'if no question mark,
    GOSUB PROCESS.ONE.COMMAND    'one command entered
ELSE
    GOSUB PROCESS.ONE.QUERY
END IF
'
EXIT SUB
'
'*************************************************
' sending a waveform to the 2712 is a different    *
' process from all others, so do it separately.    *
'*************************************************
'
PROCESS.ONE.COMMAND:
'
'if a curve command, then go and send it
IF LEFT$(wrt$, 3) = "CUR" OR_
    LEFT$(wrt$, 5) = "CURVE" THEN
        GOSUB SEND.WAVE
```





```
ELSE
    func% = 3                    'else send all other
    CALL RS232.CALLS             'commands the same way
    rd$ = LEFT$(rd$, LEN(rd$) - 2)
    PRINT rd$;
END IF
'
RETURN
'*************************************************
' since receiving a response from a curve query is *
' different, the program distinguishes this query  *
' from the rest and processes it separately.       *
'*************************************************
'
PROCESS.ONE.QUERY:
'
IF LEFT$(wrt$, 4) = "CUR?" OR_
    LEFT$(wrt$, 6) = "CURVE?" OR_
    LEFT$(wrt$, 5) = "CURV?" THEN
    GOSUB ACQUIRE.WAVE
ELSE                             'if length of response
    func% = 3                    'too large to fit in
    CALL RS232.CALLS             'the window defined use,
    IF LEN(rd$) > 66 THEN        'this routine to display
        DO WHILE LEN(rd$) >= 3   'it in groups small
            GOSUB FRAGMENT.RESPONSE    'enough to fit
        LOOP
    ELSE
        PRINT rd$;
    END IF
END IF
'
RETURN
'
ACQUIRE.WAVE:
'
'================================
' insure a binary waveform transfer
'================================
hold.wrt$ = wrt$
'
' waveform preamble for binary data transfer
'
wrt$ = "WFM ENC:B;"
func% = 3
CALL RS232.CALLS
 wrt$ = hold.wrt$
```

```
'
' validate that curve query is properly formatted
'
IF INSTR(wrt$, "CURVE") OR INSTR(wrt$, "CURV") THEN
    IF LEN(wrt$) < 8 THEN
        posit = INSTR(wrt$, "?")
        tem.wrt$ = LEFT$(wrt$, posit)
        wrt$ = tem.wrt$ + " D"
    END IF
ELSEIF LEN(wrt$) < 6 THEN
    posit = INSTR(wrt$, "?")
    tem.wrt$ = LEFT$(wrt$, posit)
    wrt$ = tem.wrt$ + " D"
END IF
'====================
'send the curve query
'====================
wrt$ = "HDR ON;" + wrt$        'ensure that hdr is on
PRINT #1, wrt$
'
'============================
'read and format the response
'============================
func% = 5
CALL RS232.CALLS
'
'============
'go display it
'============
'
GOSUB DISPLAY.WAVE
'
RETURN
'*************************************************
' This is an example of a hex waveform sent to the *
' 2712. The send function assumes that an acquire  *
' waveform function has been performed previously. *
' This is an arbitrary decision for demo purposes. *
' Any combination of characters can be constructed *
' to create a waveform to be sent to the 2712.     *
' Also, the waveform is always sent to storage     *
' location C. This is an arbitrary target location *
' A,B or C can receive the waveform.               *
'*************************************************
'
SEND.WAVE:
'
```





```
' ============================
' will always send to waveform C
' ============================
'
' first make sure save C is on
'
wrt$ = "SAV C:ON;"
func% = 3
CALL RS232.CALLS
'
' next, set preamble to point to target waveform (C)
'
wrt$ = "WFM WFI:C;"
func% = 3
CALL RS232.CALLS
'
' next construct the waveform to be sent
'
'curve command with argument indicating hex transfer
wrt$ = "CURVE #H0"
wrt$ = wrt$ + HEX$(513)        'byte count
FOR x% = 0 TO 511    'actual waveform data conversion
    TEMP$ = STR$(wfm%(x%))      'routine, decimal to hex
    IF VAL(TEMP$) < 16 THEN
        wrt$ = wrt$ + "0"
    END IF
    wrt$ = wrt$ + HEX$(wfm%(x%))
NEXT
'
' next calculate checksum
' NOTE: see explanation of CURVE command for
' information on checksum calculation
'
check.sum& = 3
'
FOR x% = 0 TO 511
    check.sum& = check.sum& + wfm%(x%)
NEXT
'
check.sum& = check.sum& MOD 256
check.sum& = (256 - check.sum&) MOD 256
'
IF check.sum& < 16 THEN
    wrt$ = wrt$ + "0"
END IF
'
wrt$ = wrt$ + HEX$(check.sum&)
```

```
'
' finally the waveform is ready to send
'
func% = 3
CALL RS232.CALLS
'
RETURN
 ' display up to 5 lines of response,
' 66 characters per line.
'
FRAGMENT.RESPONSE:
'
display.line% = 1
DO WHILE display.line% < 6
    rd1$ = MID$(rd$, 1, 66)
    posit = 1
    DO WHILE posit <> 0
        posit = INSTR(rd1$, CHR$(10))
        IF posit <> 0 THEN
            MID$(rd1$, posit, 1) = " "
        END IF
    LOOP
    posit = INSTR(rd1$, CHR$(13))
    IF posit <> 0 THEN
        rd1$ = LEFT$(rd1$, posit - 1)
    END IF
'
    LOCATE display.line% + 19, 8, 0
    PRINT rd1$;
    rd$ = RIGHT$(rd$, LEN(rd$) - LEN(rd1$))
'
    IF LEN(rd$) >= 3 THEN
        display.line% = display.line% + 1
    ELSE
        display.line% = 6
    END IF
LOOP
'
IF LEN(rd$) >= 3 THEN
    GOSUB TEMP.STOP
END IF
'
RETURN
'=======================================
' routine to display the data on screen
'=======================================
DISPLAY.WAVE:
```





```
x = 20: y = 8
FOR IX% = 0 TO 511
    LOCATE x, y
    PRINT wfm%(IX%);
    y = y + 5
    IF y > 70 THEN
        y = 8
        x = x + 1
    END IF
    IF x > 24 THEN
        x = 20
        GOSUB TEMP.STOP
    END IF
NEXT
RETURN
'
'=============================================
' routine invoked to control display of response
'=============================================
'
TEMP.STOP:
COLOR 0, 7
LOCATE 25, 9, 0: PRINT "Press (Enter) to continue";
DO WHILE INKEY$ <> CHR$(13)
LOOP
COLOR 9, 9
FOR indx% = 20 TO 25
    LOCATE indx%, 8
    PRINT STRING$(66, CHR$(32));
NEXT
COLOR 14, 9
RETURN
END SUB
'
```

# Appendices

Replace this page with the tab divider of the same name.

# Appendix A
# GPIB System Concepts

The General Purpose Interface Bus (GPIB) is a digital control bus that allows efficient communications between self-contained instruments or devices connected in an instrumentation system. The GPIB is an interface system independent of the stimulus or measurement functions incorporated in any instrument.

Instruments or devices designed to operate on the GPIB digital control bus must be developed according to the specifications contained in IEEE Std 488-1978, *IEEE Standard Digital Interface for Programmable Instrumentation.* The IEEE 488 digital interface is commonly known as the General Purpose Interface Bus (GPIB). This section discusses the basic concepts of the GPIB. For complete specifications, refer to the IEEE Std 488-1978 standard, published by the Institute of Electrical and Electronics Engineers, Inc.

The GPIB has four elements: mechanical, electrical, functional, and operational. Of these four, only the last is device-dependent. Operational elements state the way in which each instrument reacts to a signal on the bus.

## MECHANICAL ELEMENTS

The IEEE Std 488 defines the GPIB connector and cable assembly as the mechanical elements of the instrumentation system. Standardizing the connector and cable assembly ensures that GPIB-compatible instruments can be physically linked together with complete pin compatibility. The connector has 24 pins; sixteen active signal lines, seven interlaced grounds, and 1 shield connection. Standard connector pin arrangement and nomenclature for the digital control signals are illustrated in Figure A-1.

The cable that attaches to the GPIB connector must be no longer than 20 meters with no more than fifteen peripheral devices (including a GPIB controller) connected at one time. The interconnecting cable assembly, which is offered as an optional accessory to the spectrum analyzer, is provided with a plug and receptacle connector type at each end of the cable to allow either a star or linear bus structure. Contact your local





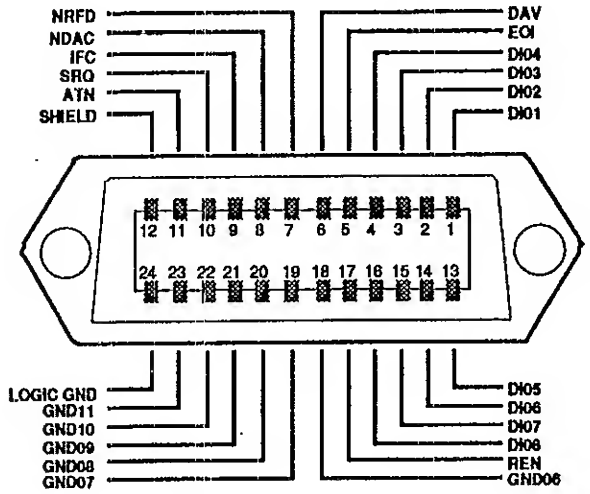



**Figure A-1. IEEE Std 488 (GPIB) Connector.**

Tektronix Field Office or representative for cable ordering information. Connectors may be rigidly stacked, using standard counter-bored captive screws.

## ELECTRICAL ELEMENTS

The voltage and current values required at the connector nodes on the bus are based on TTL technology. The power source is not to exceed +5.25 V referenced to logic ground. The standard defines the logic levels as follows:

- Logical 1 is a true state:
  low voltage level (≤+0.8 V), signal line is asserted.
- Logical 0 is a false state:
  high voltage level (≥+2.0 V), signal line not asserted.

Messages can be sent over the GPIB as either active-true or passive-true signals. Passive-true signals occur at a high voltage level and must be carried on a signal line using open-collector devices. Active-true signals occur at a low voltage level.

Table A-1. Major GPIB Interface Functions.

| Interface Function | Symbol |
|---|---|
| Source Handshake | SH |
| Acceptor Handshake | AH |
| Talker or Extended Talker | T or TE |
| Listener or Extended Listener | L or LE |
| Service Request | SR |
| Remote-Local | RL |
| Parallel Poll | PP |
| Device Clear | DC |
| Device Trigger | DT |
| Controller | C |

## FUNCTIONAL ELEMENTS

The functional elements of the GPIB cover three areas:

- The ten major interface functions of the GPIB are listed in Table A-1. Each interface function is a system element that provides the basic operational facility through which an instrument can receive, process, and send messages over the GPIB.
- The second functional element is the specific protocol by which the interface functions send and receive their limited set of messages.
- The logical and timing relationships between allowable states for all interface functions is the third area covered.

Note that while the IEEE Std 488 standard defines the ten interface functions, the specific protocol, and timing relationships, not every instrument on the bus will have all ten interface functions incorporated. Only those functions important to a particular instrument's purpose need to be implemented.





## A TYPICAL GPIB SYSTEM

A typical GPIB instrumentation system is shown in Figure A-2, and it includes the nomenclature for the sixteen active signal lines. Only four instruments are shown in this example, but the GPIB can support up to fifteen instruments connected directly to the bus. However, more than fifteen devices can be interfaced to a single bus if they do not connect directly to the bus, but are interfaced through a primary device. Such a scheme can be used for programmable plug-ins housed in a mainframe where the mainframe is addressed with a primary address code and the plug-ins are addressed with a secondary address code.

To maintain the electrical characteristics of the bus, a device load should be connected for each two meters of cable length. Although instruments are usually spaced no more than two meters apart, they can be separated farther apart if the required number of device loads are lumped at any given point. For proper operation, at least two-thirds of the instruments connected directly to the bus must be in the power-on state.

## TALKERS, LISTENERS, AND CONTROLLERS

A talker is an instrument that can send messages and data over the bus. A listener is an instrument that can accept messages and data from the bus. An instrument can be a talker only, listener only, or be both a talker and a listener. Unless a device is in the talk-only or listen-only mode, it can only communicate with other devices on the bus when it is enabled to do so by the controller in charge of the instrumentation system.

A controller is an instrument that determines, by software routines, which instrument will talk and which instruments will listen during any given time interval. The controller has the ability to assign itself as a talker or a listener whenever the program routine requires it. In addition to designating the current talker and listeners for a particular communication sequence, the controller is assigned the task of sending special codes and commands (called interface control messages) to any or all instruments on the bus. A complete operating system may contain more than one controller. The IEEE standard has provisions for a system controller that operates with another controller in charge of the bus. The controller that is in charge of the bus can take control only when it is directed to do so by the system controller. The system controller may be, but is not necessarily, the controller in charge of the bus.

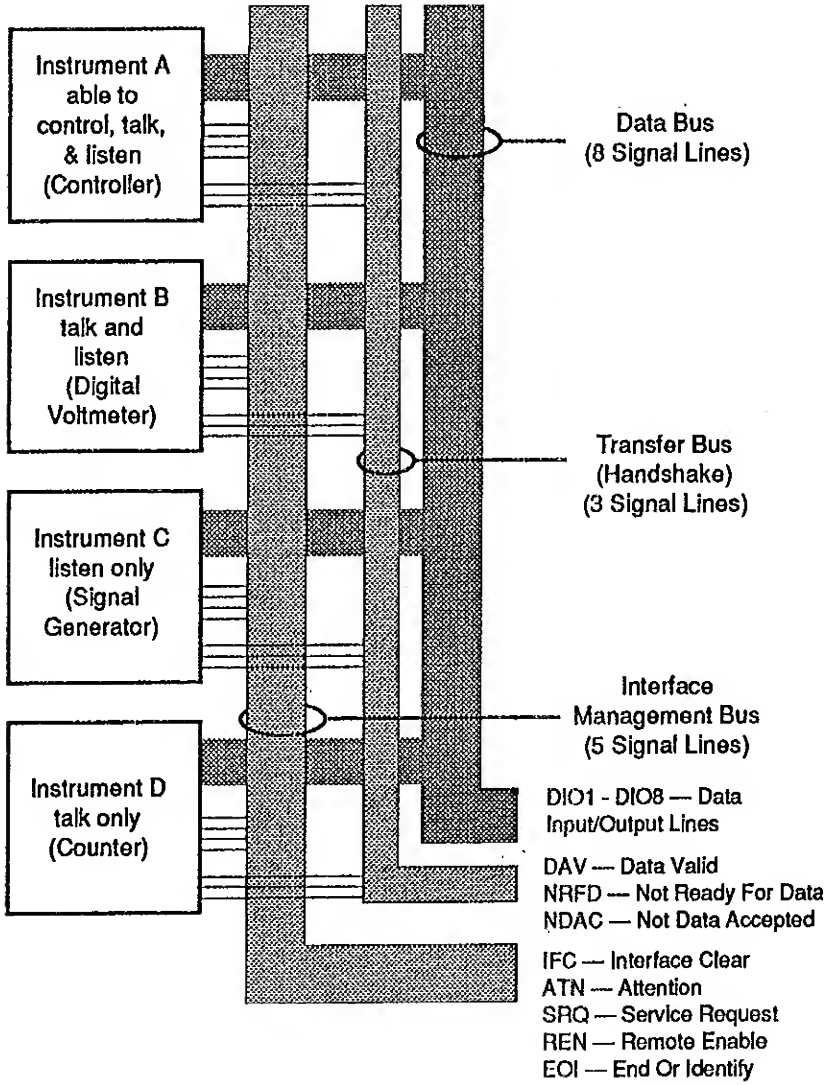



**Figure A-2. Typical GPIB System.**





## INTERFACE CONTROL MESSAGES

The two types of interface control messages are multi-line messages sent over the data bus and uni-line messages. A message that shares a group of signal lines with other messages, in some mutually exclusive set, is called a multi-line message (only one multi-line message [message byte] can be sent at one time). A message sent over a single line is called a uni-line message (two or more of these messages can be sent concurrently.)

Only multi-line messages are discussed here; uni-line messages are discussed later in this section; see ***GPIB Signal Line Definitions.***

The interface control messages (Figure A-3) are sent and received over the data bus only when the ATN (attention) line is asserted (true). Interface message coding can be related to the ISO (International Standards Organization) 7-bit code by relating data bus lines DIO1 through DIO7 to bits B1 through B7, respectively, in the Bits column in Figure A-3.

Interface control messages (Table A-2) include the primary talk and listen addresses for instruments on the bus, addressed commands (only instruments previously addressed to listen will respond to these commands), universal commands (all instruments, whether they have been addressed or not, will respond to these commands), and secondary addresses for devices interfaced through a primary instrument. Parallel Poll Enable (PPE) messages are derived from the characters in the first column under Lower Case letters in Figure A-3 (decimal coded characters 96 through 111). The standard recommends the use of decimal code 112 (lower case letter p) for the Parallel Poll Disable (PPD) command. All parallel poll-configured instruments respond with status information at the same time when the EOI line is asserted and ATN is true.

## DEVICE-DEPENDENT MESSAGES

The IEEE standard does not specify the coding of device-dependent messages; messages that control the internal operating functions of a device. After addressing a talker and the required number of listeners via interface control messages, the controller unasserts the ATN line (false) on the bus. When ATN becomes false (high), any commonly understood 8-bit binary code may be used to represent a device-dependent message.

The standard recommends that the alphanumeric codes associated with the numbers, symbols, and upper case characters (decimal 32 to decimal 94) in the ASCII Code Chart (Figure A-3) be used to compose device-dependent messages. One example of a device-dependent message could be the following ASCII character string that controls the signal generator from Figure A-2:

```
MODE V;VOLTS 2.5E-3;FREQ 1.0E3
```

The ASCII character string from this example, sent when the ATN line is unasserted, tells the signal generator to set its front panel controls to the voltage mode (`MODE V;VOLTS`) and produce a 2.5 mV signal (`2.5E-3;`)at a frequency of 1000 Hz (`FREQ 1.0E3`).

When 8-bit binary codes other than the ISO 7-bit are used for device-dependent messages, the most significant bit should be on data line DI08 (for bit-8).

To summarize the difference between interface control messages and device-dependent messages on the data bus, remember that any message sent or received when the ATN line is asserted (low) is an interface control message. Any message (data bytes) sent or received when the ATN line is unasserted (high) is a device-dependent message.





**Table A-2. Interface Messages and Functions: Remote Messages Sent.**

| Mnemonic | Message | Function |
|---|---|---|
| ATN | Attention | AH,C,L,LE,PP,SH,T,TE |
| DAC | Data Accepted | SH |
| DAV | Data Valid | AH |
| DCL[a] | Device Clear | DC |
| GET[a] | Group Execute Trigger | DT |
| GTL[a] | Go To Local | RL |
| IFC | Interface Clear | C,L,LE,T,TE |
| LLO[a] | Local Lockout | RL |
| MSA[a] | My Secondary Address | LE,TE |
| MTA[a] | My Talk Address | T,TE |
| PPC[a] | Parallel Poll Configure | PP |
| PPD[a] | Parallel Poll Disable | PP |
| PPE[a] | Parallel Poll Enable | PP |
| PPU[a] | Parallel Poll Unconfigure | PP |
| REN | Remote Enable | RL |
| RFD | Ready For Data | SH |
| SDC[a] | Selected Device Clear | DC |
| SPD[a] | Serial Poll Disable | T,TE |
| SPE[a] | Serial Poll Enable | T,TE |
| SRQ | Service Request | (via C) |
| TCT[a] | Take Control | C |
| UNL[a] | Unlisten | L,LE |

[a] Multi-line messages.

**Table A-2. Interface Messages and Functions: Remote Messages Sent (Continued).**

| Mnemonic | Message | Function |
|---|---|---|
| ATN | Attention | C |
| DAC | Data Accepted | AH |
| DAV | Data Valid | SH |
| DCL[a] | Device Clear | (via C) |
| GET[a] | Group Execute Trigger | (via C) |
| GTL[a] | Go To Local | (via C) |
| IFC | Interface Clear | C |
| LLO[a] | Local Lockout | (via C) |
| MSA[a] | My Secondary Address | (via C) |
| MTA[a] | My Talk Address | (via C) |
| PPC[a] | Parallel Poll Configure | (via C) |
| PPD[a] | Parallel Poll Disable | (via C) |
| PPE[a] | Parallel Poll Enable | (via C) |
| PPU[a] | Parallel Poll Unconfigure | (via C) |
| REN | Remote Enable | C |
| RFD | Ready For Data | AH |
| SDC[a] | Selected Device Clear | (via C) |
| SPD[a] | Serial Poll Disable | (via C) |
| SPE[a] | Serial Poll Enable | (via C) |
| SRQ | Service Request | SR |
| TCT[a] | Take Control | (via C) |
| UNL[a] | Unlisten | (via C) |
| UNT[a] | Untalk | (via C) |

[a] Multi-line messages.





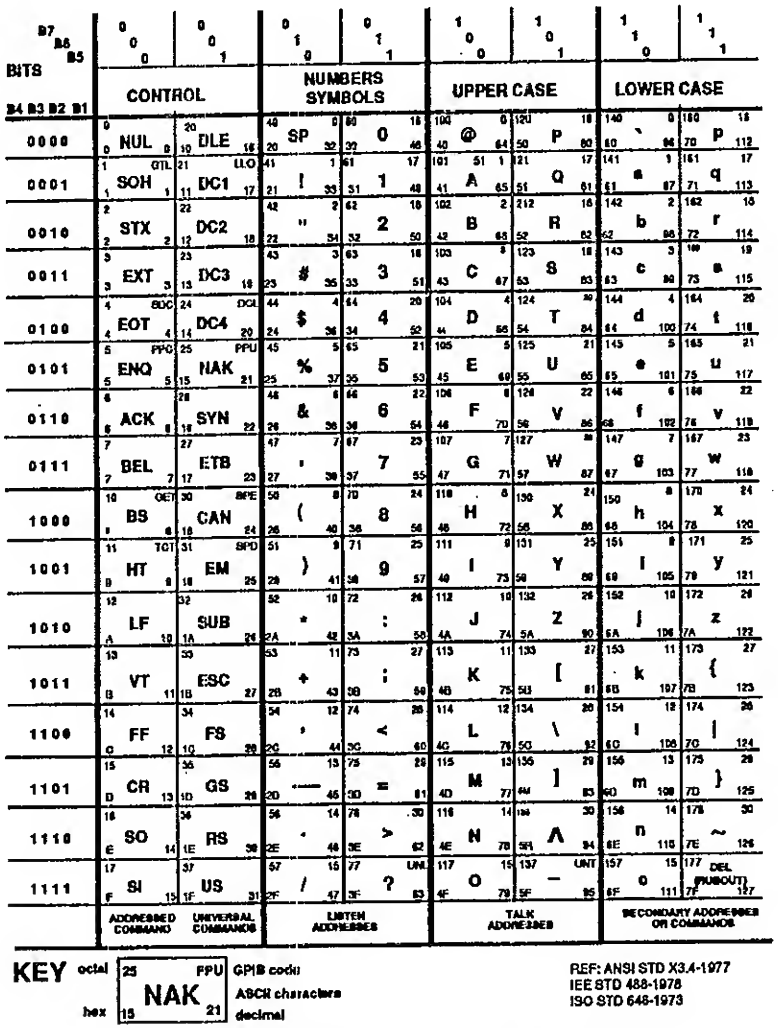



**Figure A-3. ASCII and GPIB Code Chart.**

## GPIB SIGNAL LINE DEFINITIONS

Figure A-2 shows how the sixteen active signal lines on the GPIB are functionally divided into three component buses: an 8-line data bus, a 3-line data byte transfer control (handshake) bus, and a 5-line general interface management bus.

The data bus contains eight bidirectional signal lines, DI01 through DI08. Information in the form of data bytes is transferred over this bus. A handshake timing sequence between the enabled talker and the enabled listeners on the three-line data transfer control bus transfers one data byte (eight bits) at a time. These data bytes are sent and received in a byte-serial, bit-parallel fashion.

Since the handshake sequence is an asynchronous operation (no clock signal on the bus), the data transfer rate is only as fast as the slowest instrument involved in a data byte transfer. A talker cannot place data bytes on the bus faster than the slowest listener can accept them.

Figure A-4 illustrates the flow of data bytes on the bus when a typical controller sends ASCII data to an assigned listener. The first data byte (decimal 44) enables an instrument at address 12 as a primary listener. The second data byte (decimal 108) is optional; for example, enabling a plug-in device at secondary address 12 as the final destination of the data to follow. The data is the two ASCII characters A and B (decimal 65 and decimal 66). Note that the ATN line is asserted for the first two data bytes and unasserted for the device-dependent character to indicate the last data byte in the message.

To complete the sequence, the controller activates the ATN line again and sends the universal unlisten (UNL) and untalk (UNT) commands to clear the bus. Six handshake cycles on the data transfer control bus are required to send the six data bytes.





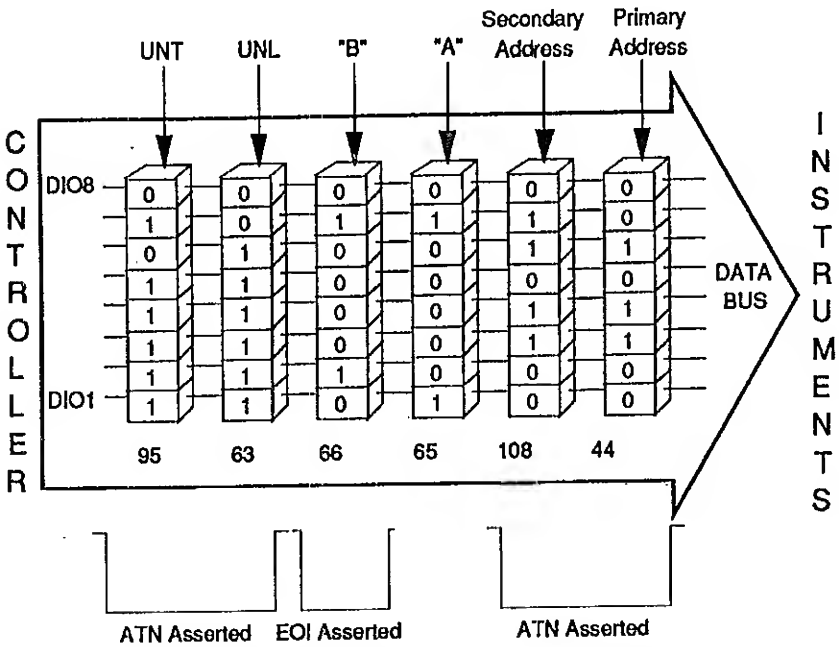



**Figure A-4. Example of Data Byte Traffic.**

### Transfer Bus (Handshake)

Each time a data byte is transferred over the data bus, an enabled talker and all enabled listeners execute a handshake sequence via signal lines DAV, NRFD, and NDAC (see Figure A-5 — the ATN line is shown to illustrate the controller's role in the process).

**DAV (Data Valid).** The DAV signal line is asserted by the talker after the talker places a data byte on the data bus. When asserted (low), DAV tells each assigned listener that a new data byte is on the bus. The talker is inhibited from asserting DAV as long as any listener holds the NRFD signal line asserted.

**NRFD (Not Ready For Data).** An asserted NRFD signal line indicates one or more of the assigned listeners are not ready to receive the next data byte from the talker. When all of the assigned listeners for a particular data byte transfer have

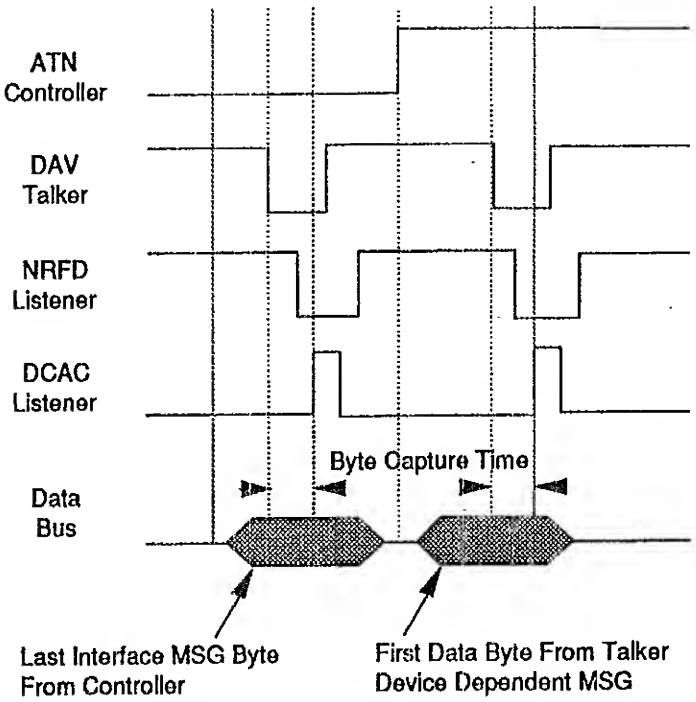



**Figure A-5. Handshake Timing Sequence, Idealized.**

released NRFD, the NRFD line becomes unasserted (high). When NRFD goes high, the RFD message (Ready For Data) tells the talker it may place the next data byte on the data bus.

**NDAC (Not Data Accepted).** Each assigned listener holds the NDAC signal line asserted until the listener accepts the data byte currently on the bus. When all assigned listeners have accepted the current data byte, the NDAC signal line becomes unasserted (high) telling the talker to remove the data byte from the bus. The DAC message (Data Accepted) tells the talker that all assigned listeners have accepted the current data byte.

Note that one handshake cycle transfers one data byte. The listeners then must reset the NRFD line high and the NDAC line low before the talker asserts DAV for the next data byte transfer. Both NRFD and NDAC high at the same time is an invalid state on the bus.





## Management Bus

The management bus is a group of five signal lines that are used to control the operation of the IEEE Std 488 (GPIB) Digital Interface.

## IFC (Interface Clear)

The system controller is the only instrument on the bus allowed to assert IFC. IFC is asserted for greater-than 100 µs to place all instruments in a predetermined state. While IFC is being sent, only the DCL (Device Clear), LLO (Local Lockout), PPU (Parallel Poll Unconfigure), and REN (Remote Enable) interface messages (universal commands) will be recognized.

## ATN (Attention)

The controller in charge is the only instrument on the bus allowed to assert ATN. ATN is asserted when an instrument connected to the bus is being enabled as a talker or listener, or when sending other interface control messages. As long as the ATN line is asserted (low), only instrument address codes and interface control messages are sent over the bus. When the ATN line is unasserted, only those instruments enabled as a talker and listener can send and receive data over the bus.

## SRQ (Service Request)

Any instrument connected to the bus can request the controller's attention by asserting the SRQ line. The controller responds by asserting ATN and executing a serial poll routine to determine which instrument is requesting service. The instrument requesting service responds with a device-dependent status byte with bit seven asserted.

When the instrument requesting service is found, program control is transferred to a service routine for that instrument. When the service routine is completed, program control returns to the main program. (The controller does not have to see the SRQ line asserted to perform a polling routine; it may do so whenever a program requires it).

## REN (Remote Enable)

The system controller asserts the REN signal line whenever the interface system operates under remote program control. The REN signal causes an instrument on the bus to select between two alternate sources of programming data. It is used with other interface control messages such as LLO (Local Lockout) or GTL

(Go To Local). A remote-local interface function indicates to an instrument that the instrument will use either information input from the interface (remote) or information input by the operator via the front panel controls (local).

### EOI (End Or Identify)

A talker can use the EOI signal line to indicate the end of a data transfer sequence. The talker asserts EOI as the last byte of data is transmitted. In this case, the EOI line is essentially a ninth data bit and must observe the same settling time as the data on the data bus.

When an instrument controller is listening, it assumes that a data byte sent with EOI asserted is the last data byte in the complete message. When the instrument controller is talking, it may assert the EOI signal line as the last data byte is transferred. The EOI line is also asserted when the ATN line is true if the controller conducts a parallel polling sequence on the bus. The EOI line is not used for a serial polling sequence.

## INTERFACE FUNCTIONS AND MESSAGES

The ten major interface functions listed in Table A-1 provide a variety of capabilities and options for an instrumentation system. These functions may be implemented in, or for, any particular instrument with instrument hardware or with a programming routine (software).

Only those functions necessary for an instrument's purpose must be implemented by the instrument's designer. An instrument will seldom have all ten interface functions. For example, an instrument generally does not need to implement the Parallel Poll (PP) function if the instrument can respond to a serial polling sequence from the controller in charge of the GPIB system.

The interface functions and their relationship to the interface control messages in Figure A-3 are discussed below. All interface control messages discussed are sent and received over the GPIB when the ATN line is asserted (low).





### RL (Remote-Local Function)

The RL function provides an instrument with the capability to select between two sources of input information. This function indicates to the instrument that its internal device-dependent functions are to respond to information input from the front panel (Local) or to corresponding programming information from the GPIB (Remote). Only the system controller is permitted to assert the REN (Remote Enable) line, whether or not it is the controller in charge at the time.

When the system controller asserts the REN line, an instrument on the GPIB goes to a remote mode when it is addressed as a listener with its listen address. An instrument remains in a remote mode until the REN line is released (high), or an optional front-panel switch on the instrument is activated to request the local mode, or a GTL (Go To Local) command is received while the instrument is enabled as a listener.

The controller can also disable the instrument's front-panel "return to local" switch(es) by sending a LLO (Local Lockout) command. The LLO command must be preceded or followed by a listen address (MLA) to cause the instrument to go to a remote mode with front-panel lockout. The UNL (Unlisten) command does not return an instrument to the local mode.

When the REN line goes false it must be recognized by all instruments on the bus, and they must go to the local mode within 100 µs. If data bytes are still being placed on the bus when REN goes false, the system program should insure that the data bytes are sent and received with the knowledge that the system is in a local mode as opposed to remote mode.

### T/TE and L/LE (Talker and Listener Functions)

The T/TE and L/LE functions are independent of each other, although they are discussed under one heading.

The T (Talker) and TE (Talker Extended) functions provide an instrument and its secondary devices, if any, with the capability to send device-dependent data over the GPIB (or, in case of a controller, the capability to send device-dependent program data) over the GPIB. The Talker (T) function is a normal function for a talker and uses only a one-byte primary address code called MTA (My Talk Address). The Talker Extended (TE) function requires a two-byte address code; an MTA code followed by the second byte called MSA (My Secondary Address).

Only one instrument in the GPIB system can be in the active talker state at any given time. A non-controller commences talking when ATN is released and continues its talker status until an Interface Clear (IFC) message occurs or an Untalk (UNT) command is received from the controller in charge. The instrument will stop talking and listen any time the controller in charge asserts ATN.

One or more instruments on the bus can be programmed for the L (Listener) function by using their specific primary listen address (called MLA). Some of the instruments interfaced to the bus may be programmed for the LE (Listener Extended) function, if implemented. The LE function requires a two-byte address code. No L or LE function is active during the time that ATN is asserted.

An instrument may be a talker only, a listener only, or implement all functions. All talker and listener functions must respond to ATN within 200 ns. They must also respond to IFC in less than 100 µs. In any case, its address code has the form X10TTTTT for a talker and X01LLLLL for a listener. For instruments with both T and L functions, the T-bit binary values are usually equal to the binary value of the L bits. Before applying power to the system, the system operator sets these five least significant bits by means of an address switch on each instrument. The controller's address code may be implemented in software.

The system program, run from the controller, designates the primary talker and primary listener status of the desired instruments by coding data bits 6 and 7. These bits are set to 1 and 0, respectively, for a talker and 0 and 1, respectively, for a listener. Secondary talk and listen addresses (or commands) are represented by the controller sending both data bits (6 and 7) as a logical 1. The controller may listen to bus traffic without actually addressing itself over the bus.

## SH and AH (Source and Acceptor Handshake Functions)

The SH and AH functions are independent of each other, although they are discussed under one heading.

The SH (Source Handshake) function guarantees proper transmission of data, while the AH (Acceptor Handshake) function guarantees proper reception of data. The interlocked handshake sequence between these two functions guarantees asynchronous transfer of each data byte. The handshake sequence is performed via the NRFD, DAV, and NDAC signal





lines on the bus (see Figure A-5). Both functions must respond to ATN within 200 ns.

The SH function must wait for the RFD (Ready For Data) message plus a minimum additional delay of 2 µs before asserting DAV. This delay allows the data to settle on the data bus. If three-state drivers are used, the settling time is reduced to RFD plus 1.1 µs. Faster settling times are allowed under special conditions and warning notes in the IEEE 488 standard. The time required for the AH function to accept an interface message byte depends upon the implementation of the function.

## DC (Device Clear Function)

The DCL (Device Clear) function allows the controller in charge to "clear" any or all instruments on the bus. The controller (under program direction) asserts ATN and sends either the universal DCL (Device Clear) command or the SDC (Selected Device Clear) command.

When the DCL message is received, all instruments on the bus must clear or initialize their internal device functions. When the controller sends the SDC command, only those instruments that have been previously addressed to listen must respond. The IEEE 488 standard does not specify the settings an instrument must go to as a result of receiving the DCL or SDC command. (In general, these commands are used only to clear the GPIB interface circuits within an instrument.)

## DT (Device Trigger Function)

The DT (Device Trigger) function allows the controller in charge to start the basic operation specified for an instrument or group of instruments on the bus. The IEEE 488 standard does not specify the basic operation an instrument is to perform when it receives the GET (Group Execute Trigger) command. To issue the GET command, the controller asserts ATN, sends the listen addresses of the instruments that are to respond to the trigger, and then sends the GET message.

Once an instrument starts its basic operation in response to GET, the instrument must not respond to subsequent trigger-state transitions until the current operation is complete. Only after completing the operation can the instrument repeat its basic operation in response to the next GET message. Thus, the basic operating time is the major factor that determines how fast the instrument(s) can be repeatedly "triggered" by commands from the bus.

## C, SR, and PP (Controller, Service Request, Parallel Poll Functions)

The C (Controller) function provides the capability to send primary talk and listen addresses, secondary addresses, universal commands, and addressed commands to all instruments on the bus. The Controller function also provides the capability to respond to a service request message (SRQ) from an instrument or to conduct a parallel poll routine to determine the status of any or all instruments on the bus that have the Parallel Poll (PP) function implemented.
If an instrumentation system has more than one controller, only the system controller is allowed to assert the IFC (Interface Clear) and REN (Remote Enable) lines at any time during system operation, whether or not it is the controller in charge at the time.

If one controller requests system control from another controller, and it receives a message from another controller to send REN, the system controller must verify that the REN line remains unasserted (false) for at least 100 µs before asserting REN. The time interval that REN is asserted depends on the remote programming sequence and will vary with the program. The IFC line must be asserted for at least 100 µs.

The Controller function has specified time intervals for certain operations. For example, the execution time for parallel polling instruments on the bus cannot be less than 2 µs. If the controller is in the controller active wait state and does not receive an internal message to conduct a parallel poll, it must wait for at least 1.5 µs before going to the controller active state in order to give the NRFD, NDAC, and EOI lines sufficient time to assume their valid states.

The controller must also have a delay of at least 2 µs (1.1 µs for tri-state drivers) so the instruments can detect that the ATN line is asserted before the controller places the first data byte on the bus.

## Taking Control (Asynchronous or Synchronous)

All data bytes transmitted over the GPIB with the ATN line asserted are interpreted as system control information. Asserting ATN directly at any moment is an asynchronous operation with respect to the bus and may cause loss of data if a handshake cycle is in progress. To prevent loss of data, a controller can take control synchronously, that is, it can monitor the Transfer Bus and only assert ATN when DAV is unasserted (false).





As a controller in charge, the system controller (program) may pass control to any other instrument in the system capable of acting as a controller. The controller in charge first addresses the other controller as a talker and then sends the TCT (Take Control) command. The other controller then becomes the controller in charge when ATN is released.

## Performing a Serial Poll

The controller-in-charge may conduct a serial poll at any time, whether or not an instrument on the bus has asserted the SRQ line. Most, but not all, instruments have the Service Request (SR) function.

To perform a serial poll, the controller first asserts ATN and issues the Untalk (UNT) and Unlisten (UNL) commands. The controller then sends the Serial Poll Enable (SPE) command, followed by the talk address of the first instrument to be polled. Next the controller releases ATN and the addressed talker responds by sending its status byte over the bus. If the addressed talker has requested service, it must assert bit seven of the status byte and encode the remaining seven bits to indicate the reason for asserting SRQ.

Status bytes are device-dependent and are not specified in the IEEE 488 standard. An addressed instrument will release its SRQ line when serially polled, but other instruments may still hold it asserted. When the controller has read the status byte of an addressed instrument, it reasserts ATN and addresses the next instrument to talk, then releases ATN and receives the instrument's status byte. The routine continues until the controller no longer detects the SRQ line asserted. At this time the controller should send the Serial Poll Disable (SPD) message and, optionally, send the UNT message to release the last active talker.

## Performing A Parallel Poll

The Parallel Poll (PP) function provides an instrument with the capability to present one, and only one, bit of status information to the controller without being previously addressed to talk. The parallel polling capability requires a commitment by the system program to periodically conduct a parallel poll sequence.

When an instrument responds to a parallel poll, the single data bit presented to the controller may or may not indicate a need for service. If the data bit is used as a service request indication, the controller should perform a serial poll in order to obtain a

complete status byte with more information (if the device has the SR function implemented).

Before an instrument can respond to a parallel poll, the GPIB system must first be configured. In a typical sequence, the controller first sends an UNL command to clear the bus of listeners, then the listen address of the device to be configured. Following this, the controller sends the PPC (Parallel Poll Configure) command followed by a PPE (Parallel Poll Enable) message. The PPE message contains coded information that tells the selected instrument which data line will carry the PP status bit for that device. This entire sequence is repeated for each instrument to be configured.

The PPE message(s) sent by the controller has the form of X110SPPP. Bit 4 (S) is called the sense bit and the three least significant bits (PPP) represent an octal number (0 through 7) that corresponds to a specific line on the data bus that an instrument must assert if its internal status has the same value as the sense bit (S may equal 1 or 0).

The actual parallel poll takes place after each instrument has been completely configured. The concept is to have the controller receive one data byte that contains status information on all of the addressed instruments. To receive this status byte, the controller asserts EOI and the ATN line. The assertion of EOI may be coincident with ATN or later, so long as both are asserted. This may occur any time after the last PPE message. The controller then reads data bus lines while ATN and EOI are asserted to interpret the status of all selected instruments.

To conclude the parallel poll, the controller releases EOI and then ATN. The instrument(s) do not need to be reconfigured for each subsequent parallel poll. The PPU (Parallel Poll Unconfigure) command will clear all device configurations and prevent them from responding to future polls. The PPD (Parallel Poll Disable) command accomplishes essentially the same results, except that the PP function remains in the "configured" state. PPU is a universal command (all instruments) while PPD is used with PPC and becomes an addressed command (only those devices selected with PPC will accept PPD.)

# APPENDIX B
# RS-232 CONCEPTS

The first part of this appendix, ***Introduction to RS-232 Communications***, introduces RS-232 communications concepts to users who have no previous experience with the RS-232 interface. The second part, ***Implementation of the RS-232 Interface***, describes implementation details for the RS-232 interface available for the 2711 and 2712 spectrum analyzers (Option 08, RS-232 Interface).

Section 1, ***Introduction***, contains all the information needed to properly configure the 2711 and 2712 for most applications. Section 6, ***Programming***, includes an example of an interactive control program for the 2711 and 2712 spectrum analyzers. This program uses the RS-232 interface and a Personal Computer (PC) controller. If additional RS-232 communications information is needed, refer to the documents listed at the end of this appendix.

## INTRODUCTION TO RS-232 COMMUNICATIONS

As with GPIB communications, RS-232 communications follow a set of electrical, mechanical, and protocol standards. The current standard is called EIA Std RS-232-C. Many different types of devices are designed to communicate according to specifications contained in standards EIA Std RS-232-C. The RS-232 standard is very flexible, allowing for many possible implementations.

The RS-232 interface is *NOT* a bus (GPIB is a bus), therefore only one device can be connected at a time. RS-232 uses an asynchronous serial data flow instead of 8-bit parallel with byte-by-byte handshaking. RS-232 does not support device addresses or serial polling.

Both devices on an RS-232 interface, the DCE (controller) and the DTE (terminal), must be configured the same way for communications to occur successfully. To meet this requirement, communications parameters for the controller and terminal must be set independently.